# RasPi

DESIGN
BUILD
CODE

8

Get hands-on with your Raspberry Pi




# Make a VIDEO PLAYER

**PLUS**
Set up VPN servers

# Welcome



We're going to play around with two things this month: the PiTFT and Twitter. The PiTFT is a smart little touchscreen designed to fit perfectly over the top of your Pi, and we're showing you how to use it as the base for your own hand-held video player – as well as showing off another project where it's used to make a working phone. And with Twitter we're going to take a look at using bots and alarms, to put the platform to use. That's not all though, as you'll see on the next page – another really good use for the Pi is as a VPN server, which you can hook up to an existing network in order to access it no matter where you are, so long as you have the right login details. There's plenty to do… have fun!

Gavin Thomas

**Deputy Editor**

## Get inspired

Discover the RasPi community's best projects

## Expert advice

Got a question? Get in touch and we'll give you a hand

## Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

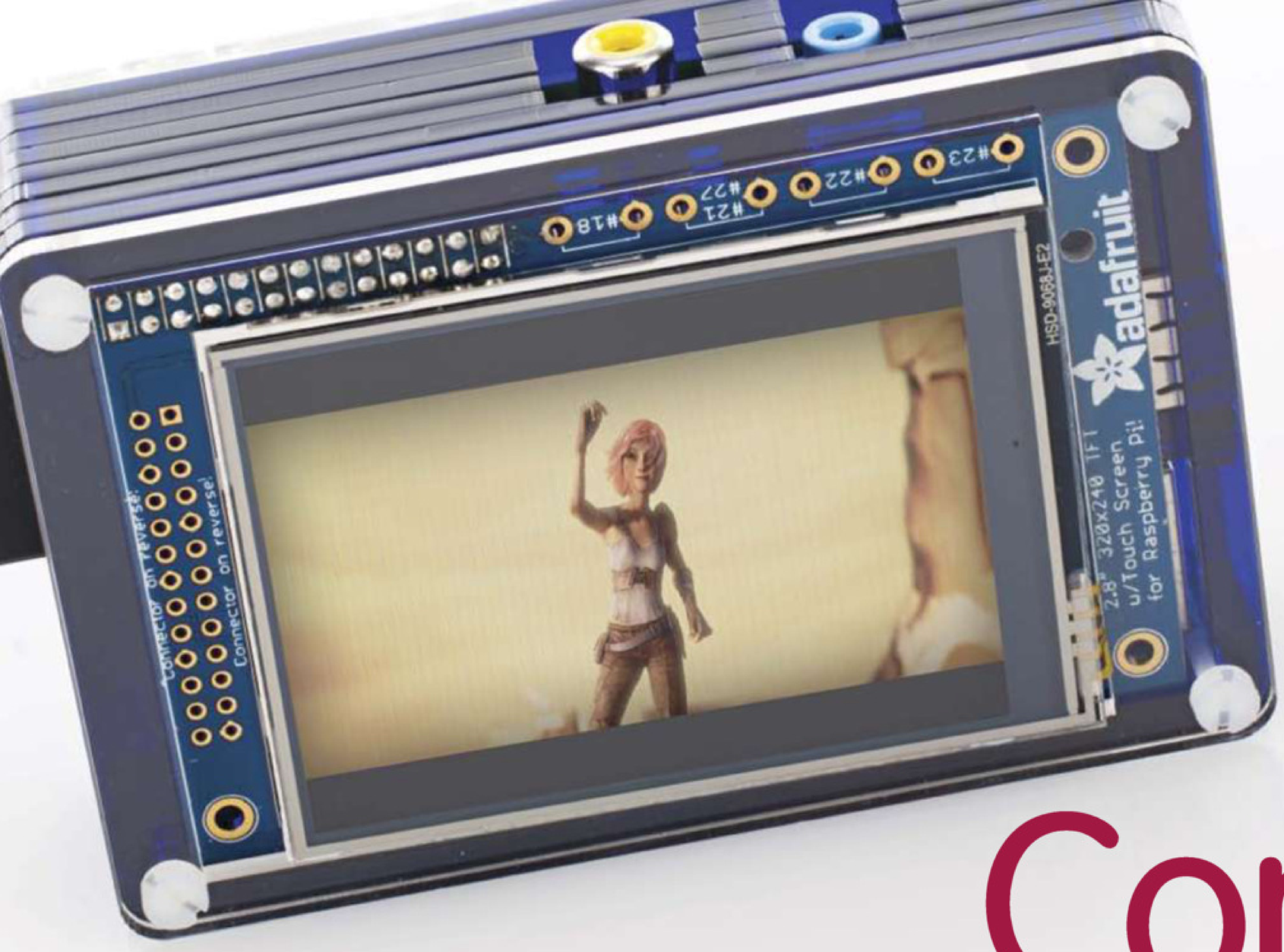


From the makers of

LinuxUser & Developer

## Join the conversation at…

@linuxusermag | Linux User & Developer | RasPi@imagine-publishing.co.uk




# Contents


## Build a Raspberry Pi video player
Play your films with a tap on the screen

## PiPhone
Turn your Pi into an mobile phone

## Code your own Twitter bot
Auto-retweet from any account you like

## Make a tweeting wireless flood sensor
...or tweak it to make a custom alarm system

## What is Raspbian?
Learn where it came from and how it's Linux

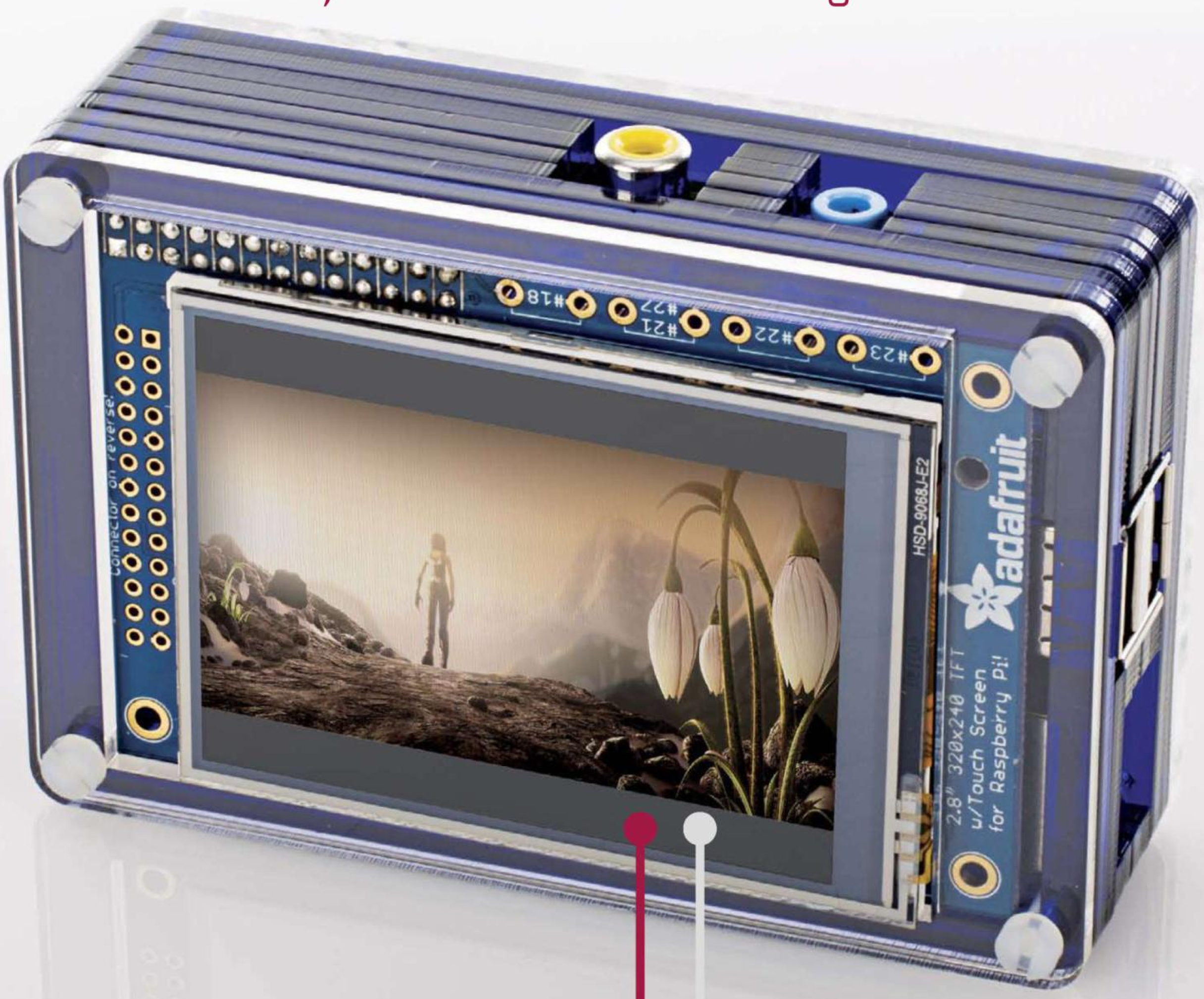
## Set up a VPN with your Pi
Remotely connect to your home network

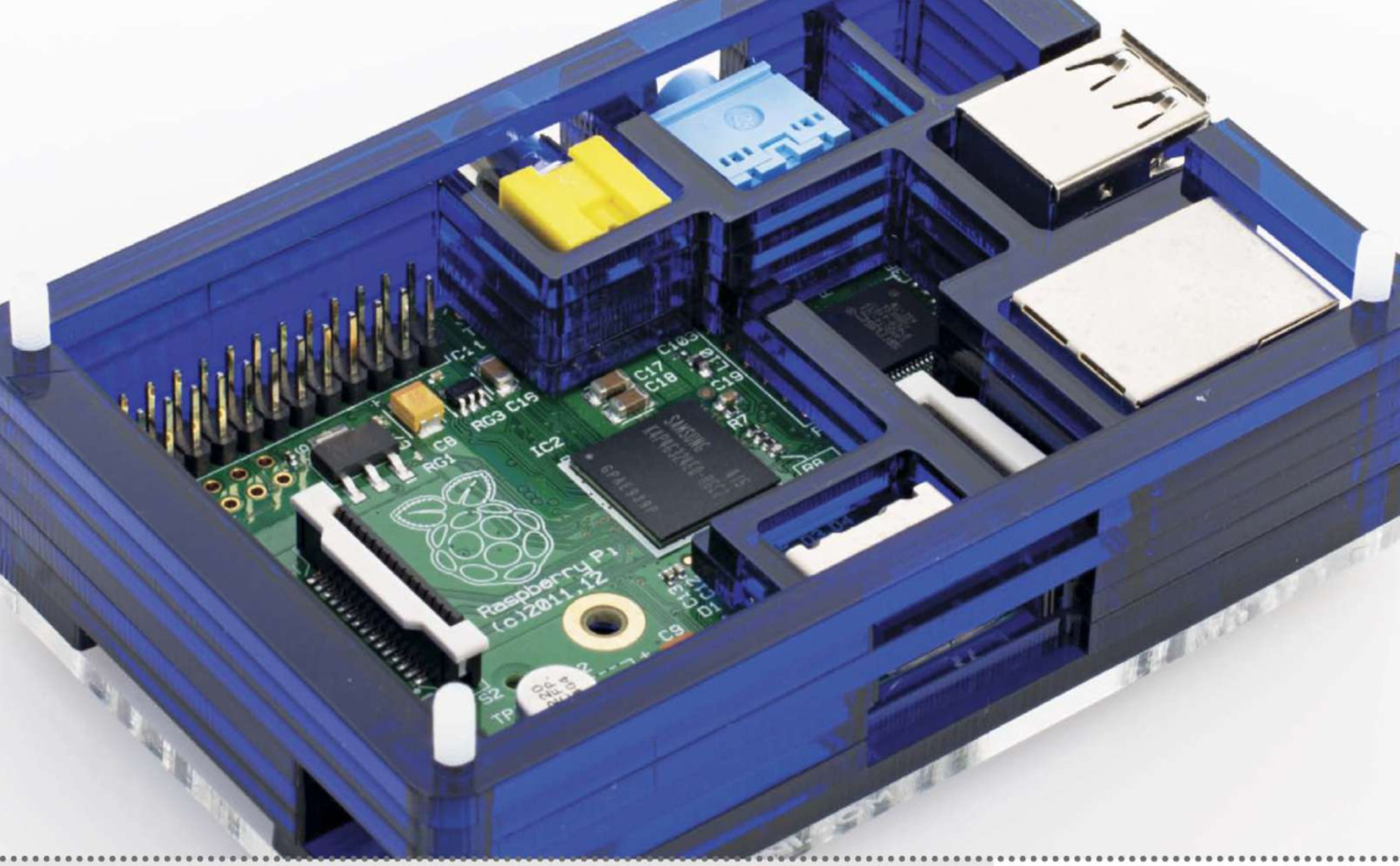
## Add vision to your Raspberry Pi
Get your Xbox's Kinect sensor involved

Storemags com

# Build a Raspberry Pi video player

Make a hand-held, touchscreen video player so you can watch films on the go






**Above** Pimoroni's Pibow case is smart, stylish and compact – ideal for this project

The Raspberry Pi is well-known for being a great choice when it comes to media projects, and if you're lucky enough to have a Raspberry Pi 2 then you'll find it even better suited to HTPC projects than ever before thanks to its upgraded processor and RAM. But you don't have to stick with RaspBMC or keep on SSHing into your Pi – if you're looking for a slightly more interesting media solution then you can always make a dedicated video player instead.

In this three-part guide we're going to show you how to set up a small touchscreen that's designed to fit perfectly over the top of your Pi. We're also going to stick the setup inside a Pibow case to make a proper hand-held device, all the wires tucked out of the way and the whole thing ready to go into your bag at a moment's notice the next time you've got a long train journey.

"Set up a small touchscreen that's designed to fit perfectly over the top of your Pi"



# Set up the PiTFT touchscreen

## This simple little touchscreen takes a bit more than just plugging in, as we show in the first part of our guide

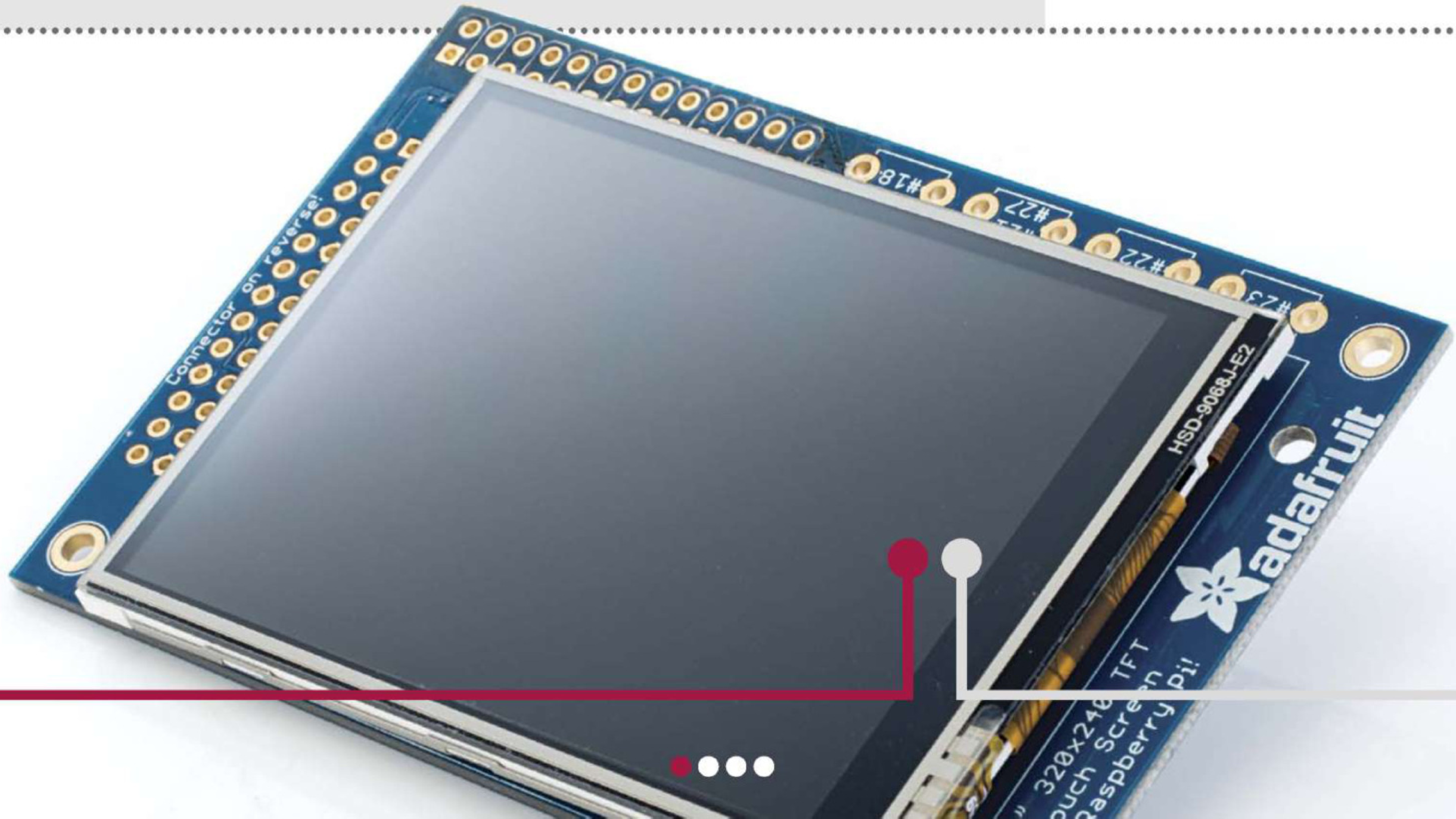
We like to talk about how portable the Raspberry Pi is, but one of the minor issues associated with this portability is the lack of a display. Carrying around a monitor is hardly practical and connecting via a phone with SSH takes some interesting trickery to get right. The PiTFT screen removes all these issues by requiring no extra power and being the size of the Raspberry Pi itself.

While Adafruit supplies a preconfigured Raspbian image for the screen, you'll need to add some extras to get it working on your current install, which is exactly what we're going to cover in this tutorial.

**THE PROJECT ESSENTIALS**

**Latest Raspbian image**
raspberrypi.org/downloads

**PiTFT**
bit.ly/1jHEJT4




**Below** The PiTFT is a 2.8-inch TFT (thin-film-transistor) LCD from Adafruit

## 01 Adafruit kernel files

The first thing we need to do is download and install the necessary kernel files. Open up a terminal or SSH in and do the following:

```
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-bin-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-dev-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-doc-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/libraspberrypi0-adafruit.deb
$ wget http://adafruit-download.s3.amazonaws.com/raspberrypi-bootloader-adafruit-112613.deb

$ sudo dpkg -i -B *.deb
```

## 02 Attach the screen

If you're using Raspbian from an image after September 2013, be sure to turn off the accelerated framebuffer using:

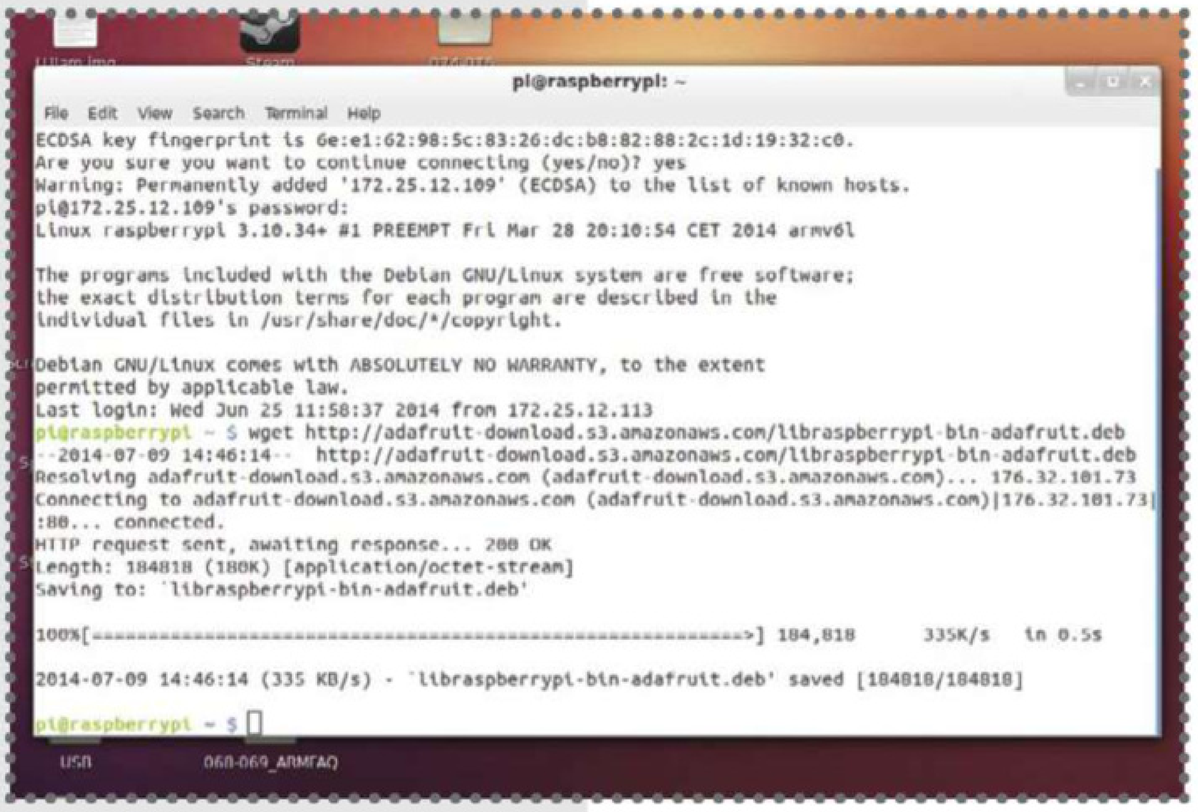
```
$ sudo mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf ~
```

After that's done, shut down the Raspberry Pi completely and plug the screen into the GPIO pins if you haven't already done so.

## 03 Install the screen driver

We can do a test to make sure that the screen works at this point. This will turn on the screen but it won't work after reboot:

"If you're using Raspbian from an image after September 2013, be sure to turn off the accelerated framebuffer"




**Below** Get the neccessary files from the Adafruit website and download them to your Pi

storemags.com

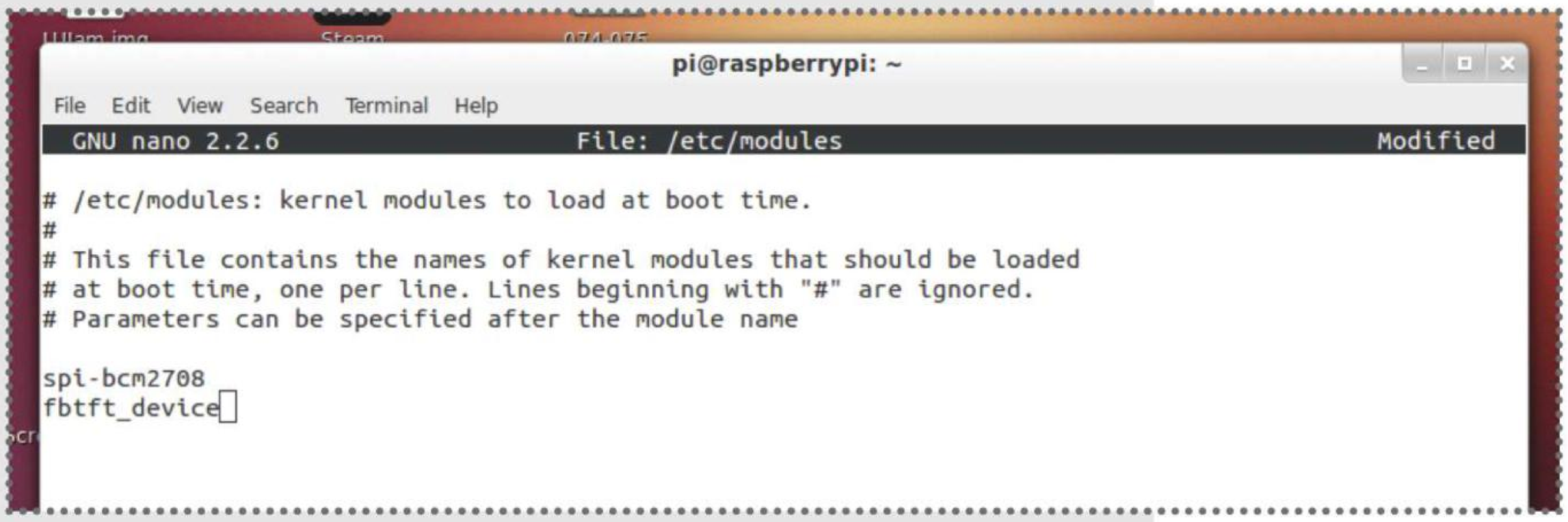
```
$ sudo modprobe spi-bcm2708
$ sudo modprobe fbtft_device name=adafruitts
rotate=90
$ export FRAMEBUFFER=/dev/fb1
$ startx
```

## 04 Add the modules

There are two modules we need to get to auto-load at boot time, including the actual PiTFT driver. Open up the modules file with the first command and add the second two lines to the list:

```
$ sudo nano /etc/modules
```

```
spi-bcm2708
fbtft_device
```




**Below** Just add the instructions after the commented-out lines

## 05 Edit configuration files

We need to create/modify the necessary configuration file so that the screen can turn on properly. The first line opens the file, while the second line needs to be added to this file:

```
$ sudo nano /etc/modprobe.d/adafruit.conf
```

```
options fbtft_device name=adafruitts
rotate=90    frequency=32000000
```

## 06 The options

You may have noticed the specific wording on these options: rotate allows you to have it at 0, 90, 180 or 270 degrees. A frequency of 32MHz gives you about 20fps on screen; if the screen is playing up, take it down to 16MHz. Now reboot the Pi.

## 07 Install touchscreen

Once back in, we need to create a new config file and populate it as shown below:

```
$ sudo mkdir /etc/X11/xorg.conf.d
$ sudo nano /etc/X11/xorg.conf.d/99-
calibration.conf

Section “InputClass”
    Identifier      “calibration”
    MatchProduct    “stmpe-ts”
    Option “Calibration” “3800 200 200 3800”
    Option “SwapAxes”     “1”
EndSection
```

## 08 Testing touch

Once that's all done you can restart X with:

```
$ FRAMEBUFFER=/dev/fb1 startx
```

You can make it so you just need to type startx by adding the following to ~/.profile:

```
export FRAMEBUFFER=/dev/fb1
```

You can also go into raspi-config to have the desktop load by default. That's it – you're now ready to start calibrating the display for your portable video player.

> “A frequency of 32MHz gives you about 20fps on screen; if the screen is playing up, take it down to 16MHz”



# Calibrate the interface

## Now your touchscreen is working, let's calibrate it and add a small interface using some simple Python code

We've got a screen with basic touch functionality but where do we go from there? Our first step is to actually make the touchscreen usable; it's resistant-style touch, so you can use your finger or a stylus with it, and this will affect the calibration of the screen once we get to it. Don't worry – if you calibrate with one method, you can always go back and recalibrate with another.

Here we're going to properly set up the touchscreen so that you can continue to use your Raspberry Pi on the go without any other devices or being within range of a decent power socket.

### THE PROJECT ESSENTIALS

**Latest Raspbian image**
raspberrypi.org/downloads

**PiTFT**
bit.ly/1jHEJT4




**Below** Do you want to use your finger or a stylus on the screen?

## 01 New rules

To make things easier for ourselves we're going to create a little rule so that we don't lose track of what the touchscreen is called in the command line. SSH in, run the next command and then copy the following text into it:

```
$ sudo nano /etc/udev/rules.d/95-stmpe.rules
SUBSYSTEM==“input”, ATTRS{name}==“stmpe-ts”,
ENV{DEVNAME}==“*event*”, SYMLINK+=“input/
touchscreen”
```

## 02 Reinstall the touchscreen

We'll need to undo what we did at the end of the last tutorial to make sure we can get the touch part working properly, by removing and then reinstalling the support for it:

```
sudo rmmod stmpe_ts; sudo modprobe stmpe_ts
```

## 03 Main event

Find out what event the touchscreen is known by with the following:

```
$ ls -l /dev/input/touchscreen
```

We can install tools now to calibrate and debug the touchscreen. Install and test them with:

```
$ sudo apt-get install evtest tslib libts-bin
$ sudo evtest /dev/input/touchscreen
```

## 04 Touch calibration

Now we can finally begin the calibration process. Enter the following so that the Pi can learn roughly where the positions are on the screen:

```
$ sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_
TSDEVICE=/dev/input/touchscreen ts_calibrate
```

“We're going to create a little rule so that we don't lose track of what the touchscreen is called in the command line”




**Left** When you draw, test for pixel accuracy and responsivity

## 05 Draw calibration

Once you've done the previous calibration, you can try out a drawing test to see how the screen reacts. Do this with the following:

```
$ sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_
TSDEVICE=/dev/input/touchscreen ts_test
```

If it's not working quite to your liking, you can try the previous step again.

## 06 Install X calibrator

Now we need to calibrate the touchscreen in X, which first means installing the correct tools. Use the following two commands to download and install the tool:

```
$ wget http://adafruit-download.s3.amazonaws.
com/xinput-calibrator_0.7.5-1_armhf.deb
$ sudo dpkg -i -B xinput-calibrator_0.7.5-1_
armhf.deb
```

Follow this up quickly by deleting irrelevant calibration data with the following:

```
$ sudo rm /etc/X11/xorg.conf.d/99-calibration.
conf
```

"Now we need to calibrate the touchscreen in X, which first means installing the correct tools"




storemags.com

## 07 X-calibrate

From your SSH shell, run startx for the screen to turn on. From here you can either try to get xinput_ calibrator typed into the terminal or type the following from the SSH session if it's easier:

```
FRAMEBUFFER=/dev/fb1 startx & DISPLAY=:0.0
xinput_calibrator
```

## 08 Save calibration

Having completed calibration, you'll get an output starting with Section "InputClass". Open up the following file and copy the output from Section to EndSection into there:

```
$ sudo nano /etc/X11/xorg.conf.d/99-
calibration.conf
```

## 09 Play around

Your new touchscreen is now properly set up! We'll go over more ways to make it easier to use in the following tutorial, but it's basically portable now and ready to have other projects, such as the PiPhone or PiCam, loaded onto it.

**Left** Be as careful as possible with the calibration to improve the accuracy

"It's basically portable now and ready to have other projects, such as the PiPhone or PiCam, loaded onto it"



# Portable Pi video player

## Need a better way to watch video on the go? Let your newly powered-up touchscreen Pi play your favourite shows

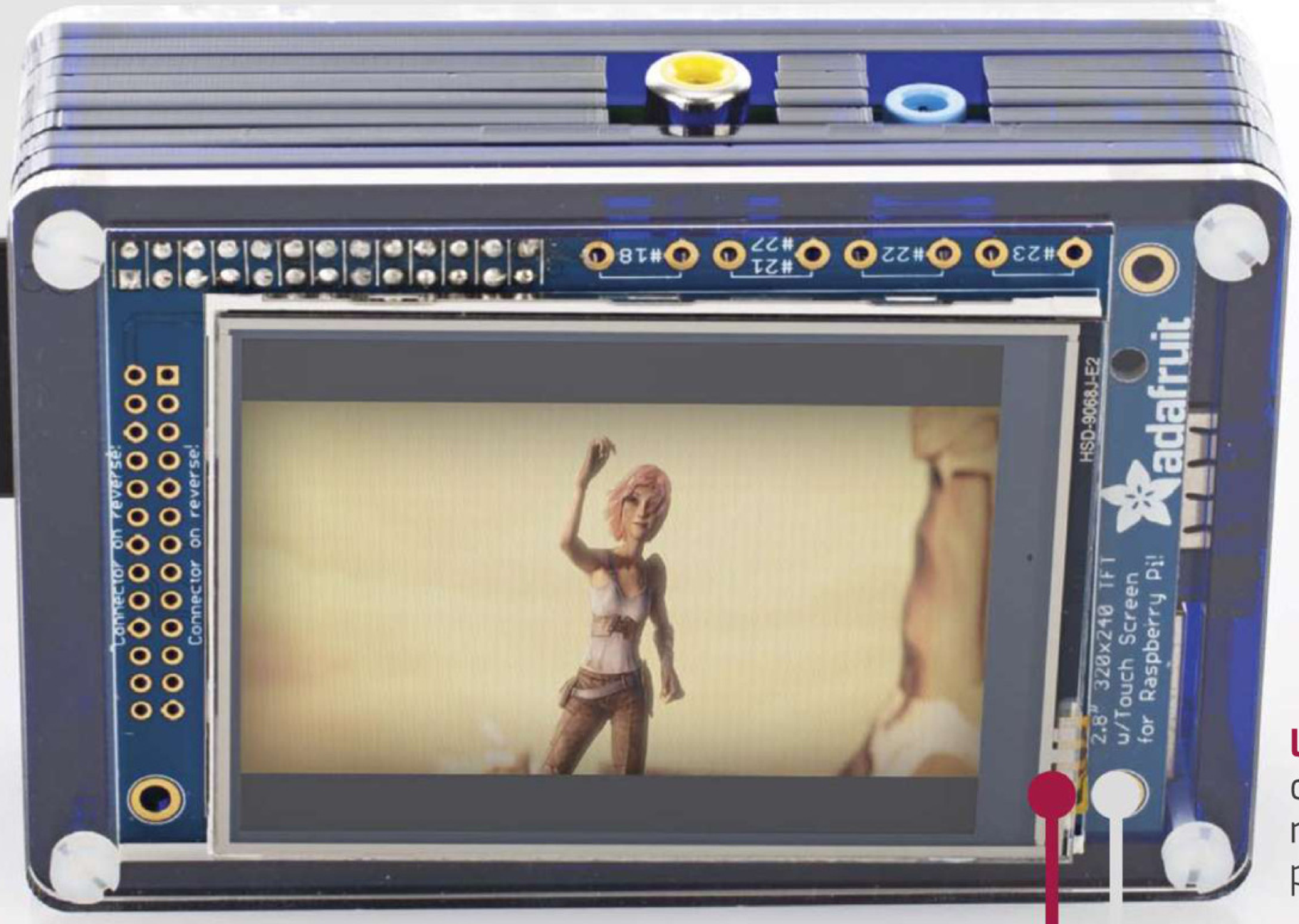
We've got a Raspberry Pi that we can make portable now – but how can you use it? The easiest thing to do with it is use it as a portable video player. It has a far superior battery to your phone and has the added bonus of being easily plugged into a TV once you reach your destination.

Before we can use it to play the videos, though, we need to make it a bit more friendly to transport and get it to play videos in a particular way. We've recommended some excellent accessories and extras to help you.

**THE PROJECT ESSENTIALS**

**Latest Raspbian image**
raspberrypi.org/downloads

**PiTFT**
bit.ly/1jHEJT4




**Left** We've kept ours simple, but you might want to add physical buttons too

## 01 Install VLC

Go to your SSH terminal and install VLC media player. It's a lightweight, highly customisable media player that is able to play just about anything you throw at it. Do this with:

```
$ sudo apt-get install vlc
```

## 02 Default settings

Here's where it gets a bit tricky, as you'll need to perform this step via the screen. Hook the Pi up to a display, mouse and keyboard or just use the mouse and keyboard on the small screen. Go to an AVI or MP4 file and right-click on it, go to Properties>Open with… and choose Customise for the next step.

## 03 Set the default

Go to Sound & Video to select VLC from the menu. Click the checkbox to set it as the default app and then press Enter to save the changes. Now when you double-click or tap on the video, it will automatically play in VLC.

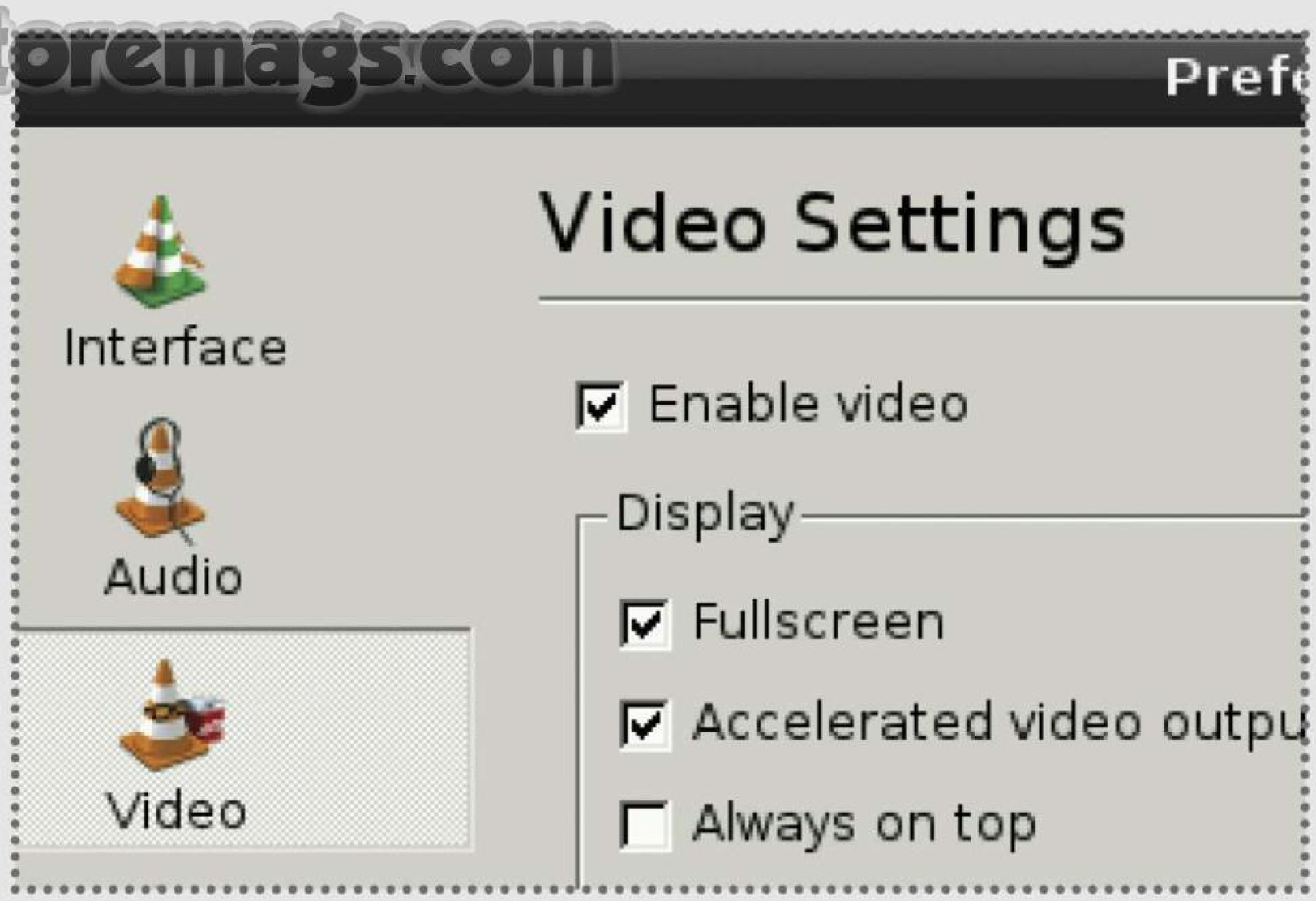



**Left** Thankfully, you'll only need to set up VLC as the default player once

"Hook the Pi up to a display, mouse and keyboard or just use the mouse and keyboard on the small screen"

storemags.com

storemags.com




**Left** Having a single instance will help keep things simpler on the small screen

## 04 VLC options

Open up VLC from the menu and right-click anywhere in the player space to open the Context menu. From here, go to Tools and then Preferences. Scroll down to click on 'Allow only one instance' and then go to the Video tab and click on the Fullscreen checkbox. Press Enter to save.

## 05 Play videos

You're done! It will play videos at the size of your Raspberry Pi screen and only one player will exist at a time. You may have better luck if you convert videos to 320x240, but it should play much larger videos just fine.

## 06 USB storage

A lot of Raspberry Pi SD cards have limited space, so it makes sense to keep your videos separate. Make sure to use storage that does not require any extra power, like a USB stick or portable hard drive. Access the storage from the File Manager, as no other mounting is required.

"You may have better luck if you convert videos to 320x240, but it should play much larger videos just fine"

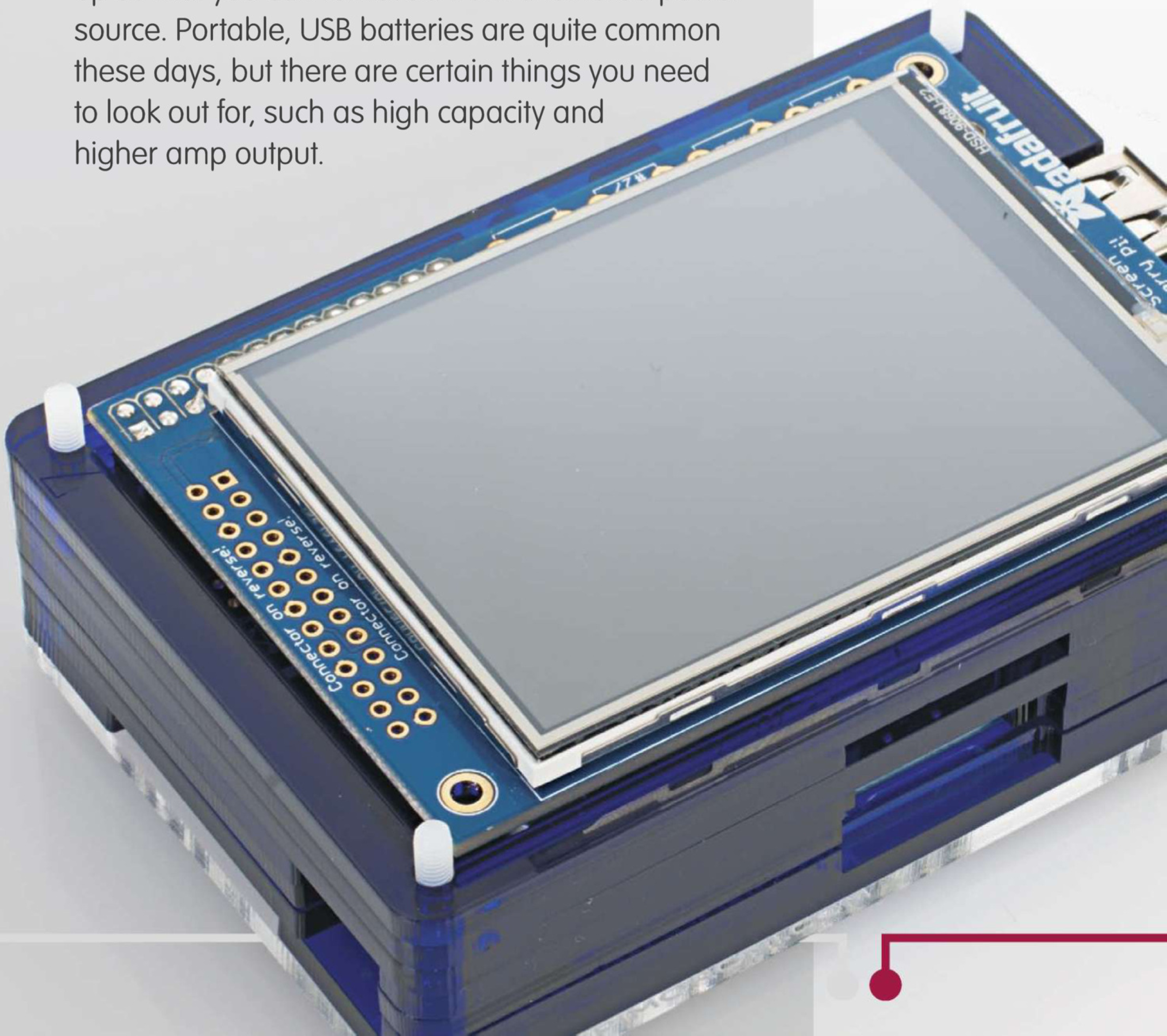
## 07 Get some protection

The PiTFT Pibow case from Pimoroni is one of the few cases – if not the only case – that houses the Raspberry Pi with the PiTFT screen. It's assembled in layers and only works with the revision 2 Raspberry Pi (the newer one with 512MB of RAM); this one won't have the pins between the yellow video port and headphone port.

## 08 Independently powered

To have it working on the go, you'll need to have it set up so that you can remove it from a tethered power source. Portable, USB batteries are quite common these days, but there are certain things you need to look out for, such as high capacity and higher amp output.

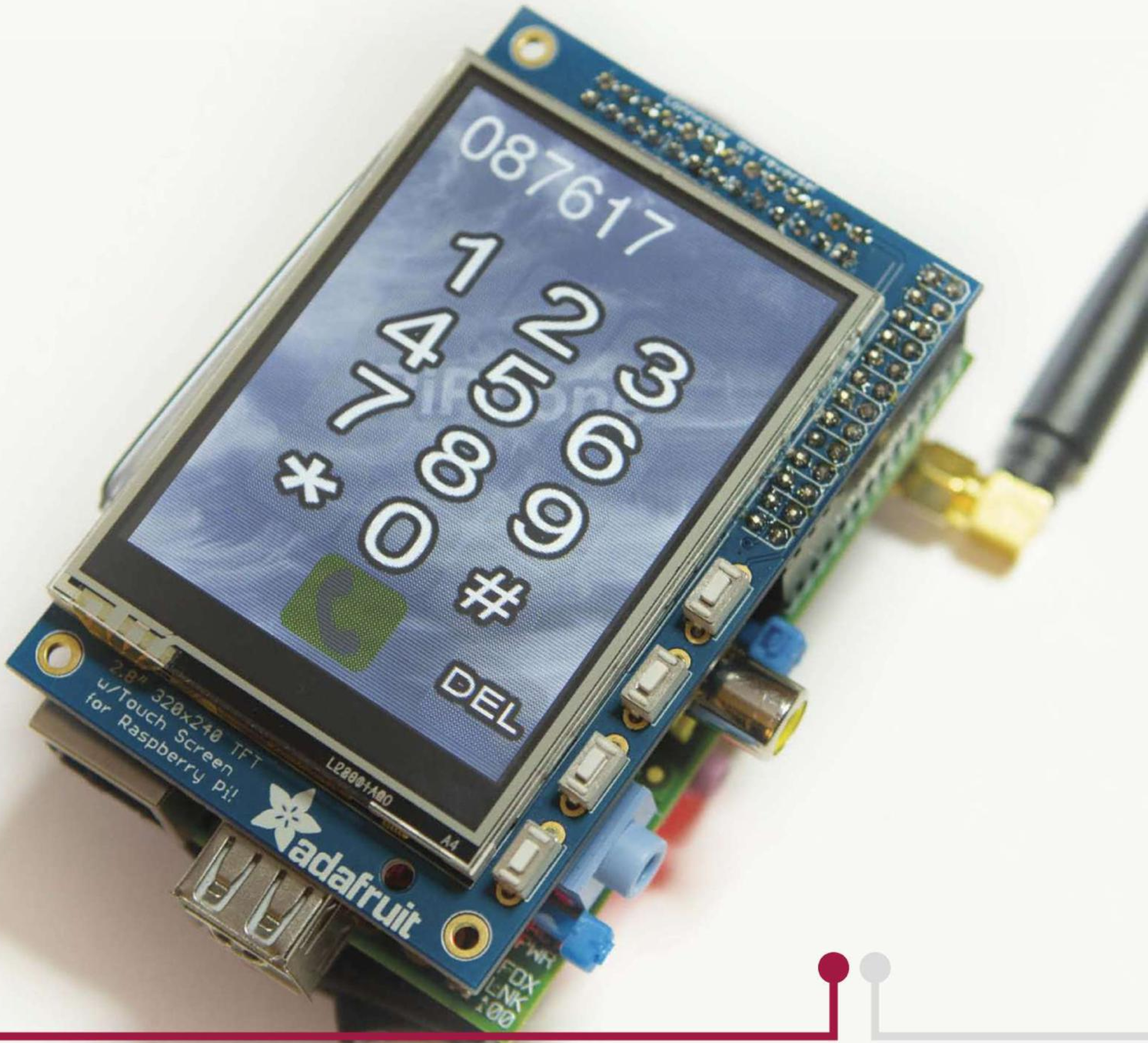
**Below** You can find everything from USB power banks to AA battery holders



storemags.com

# PiPhone

## Turning your Raspberry Pi into a mobile phone is a lot simpler than you'd think, albeit a little chunky...



### How did the idea come about for the PiPhone?

I did a presentation at a local conference a couple of weekends ago and I had to think of a topic. So I thought of the evolution of the PiPhone, where it came from. It started with the Raspberry Pi; I had a project called the Camera Pi, which was embedding a Raspberry Pi into a battery grip of an SLR camera. That went viral; it was nuts because Engadget and everywhere covered this. I'd been waiting to do that project but I couldn't do it until the Pi came out because everything else was either too expensive or too big. The Gumstix were around, and they were nice and small, but still over $200 at the time, then the Raspberry Pi came along and changed everything. So I started making with it and that got a good reaction, and I went on and started making time-lapse rails and bark-activated door openers and macro rails and all sorts of other stuff based on Raspberry Pis.

We really are living in a golden age for kids in terms of electronics: you can buy kits now that we never had – that I never had – as a kid. You can buy GPS units and GSM modules and pressure barometers, and all these kits are freely available off Adafruit and via these DIY kits. It's just a fantastic time to be living in for technology and it's great that the Raspberry Pi is building on that and encouraging kids to get these bits and pieces and put them together. Which is what I'm doing with the PiPhone: it's all just off-the-shelf parts that you can put together in a way that hadn't really been done with a Raspberry Pi. It's been done with Arduino but it never really caught on – maybe it's the name that made the PiPhone go viral? Probably!



**Dave Hunt** is an embedded systems engineer that has been making since the early Noughties, currently specialising in Raspberry Pi, Arduino and photography

**Below** The phone can currently only make calls but the code is open for modification



### Have you seen anyone else make a PiPhone yet?

The bunch of people that have said that they are building PiPhones is amazing – there are at least half a dozen people that are actually getting parts, and I don't know how many other people are thinking about it and doing it without letting people know. That's the whole idea: get people building and get people playing with Linux even if they might not be that experienced with electronics. The whole PiPhone user interface is written in 150 lines of Python – it's something that you can get your head around.

### How usable is it as an actual phone in its current state?

Not very! It basically makes calls but it doesn't receive calls. It makes calls pretty well; I've got the interface so it changes the green dial button to red when it's making a call and when you hang up it goes back to green again. It tells you when it's making a call and it establishes the call, and when you've got your headphones in you can talk and the red button will hang up the call. That part of it works quite well.

Functionality-wise it's only a few extra lines of Python to poll the GSM module for calling or listening for a string and then doing an ATA to answer the call. There's nothing at all stopping anyone from adding the extra ten lines of Python to add an answering functionality into it. I'm just waiting to see who's going to do that first – is it going to be one of the people building it? Are they going to extend it a little bit? That would be fantastic.

As for other functionality, there's very little in there. Bearing in mind it's a full Linux/X Windows system behind that Python interface.

### If you like

Interested in wireless networking and Raspberry Pis? Check out the Onion Pi project from Adafruit, where you can build a secure router: **learn.adafruit.com/onion-pi**

### Further reading

Visit Dave Hunt's blog to learn more about the PiPhone, with information on how to build it and links to the code: **bit.ly/1jNLemR**

**What projects do you have planned with the Raspberry Pi in the future?**

I've already moved on, really. I've just got myself a 3D printer and I'm ramping up on that; I've already started a project that deals with parts and motors and Raspberry Pi's driving motors and such, mounted on 3D prints in interesting ways.

I've done a time-lapse rail and I always wanted to add a panning head, so as well as moving sideways along a rail I always wanted the camera to be able to pan left, right, up and down in a rotational manner. I'm kind of working on that at the moment with parts printed on the 3D printer. It makes a lot of the tools in my shed totally obsolete – files and saws and all sorts of stuff that I would have had to use for making very complicated pieces. Now you just 3D print the part, mount a motor to it and it's there.

So you should see lots more of that coming out but it's still going to be all Raspberry Pi/Arduino-based; there's always going to be an embedded computer in it some way.

**Below** The Adafruit Learning System has a great guide to building this: **bit.ly/1Kel7Wo**




"It's just off-the-shelf parts that you can put together in a way that hadn't been done before with a Pi"

# Code your own Twitter bot

## Create your very own Twitter bot that can retweet chunks of wisdom from any account

Twitter is a useful way of sharing information with the world and it's our favourite method of giving our views quickly and conveniently. Many millions of people use the microblogging platform from their computers, mobile devices and possibly even have it on their televisions. You don't need to keep pressing that retweet button, though. With a sprinkling of Python, you can have your Raspberry Pi do it for you. Here's how to create your own Twitter bot…

**THE PROJECT ESSENTIALS**

**Latest Raspbian image**

**Internet connection**

## 01 Installing the required software

Log into the Raspbian system with the username Pi and the password raspberry. Get the latest package lists using the command `sudo apt-get update`. Then install the Python Package installer using `sudo apt-get install python-pip`. Once you've done that, run `sudo pip install twython` to install the Twitter library we'll be using.

## 02 Registering with Twitter

We need to authenticate with Twitter using OAuth. Before this, you need to go to **https://dev.twitter.com/apps** and sign in with the account you'd like your Pi to tweet from. Click the 'Create a new application' button. We called our application 'LUD Pi Bot', gave it the same description and set the Website to **http://www.linuxuser.co.uk/**. The Callback URL is unnecessary. You'll have to agree with the developer rules and then click the Create button.

## 03 Creating an access token

Go to the Settings tab and change the Access type from 'Read only' to 'Read and Write'. Then click the 'Update

"If the tweet's time is newer than the time the function was last called, we retweet it"

this Twitter application's settings' button. The next step is to create an access token. To do that, click the 'Create my access token' button. If you refresh the details page, you should have a consumer key, a consumer secret and access token, plus an access token secret. This is everything we need to authenticate with Twitter.

## 04 Authenticating with Twitter

We're going to create our bot as a class, where we authenticate with Twitter in the constructor. We take the tokens from the previous steps as parameters and use them to create an instance of the Twython API. We also have a variable, last_ran, which is set to the current time. This is used to check if there are new tweets later on.

## 05 Retweeting a user

The first thing we need to do is get a list of the user's latest tweets. We then loop through each tweet and get its creation time as a string, which is then converted to a datetime object. We then check that the tweet's time is newer than the time the function was last called – and if so, retweet the tweet.

## 06 The main section

The main section is straightforward. We create an instance of the bot class using our tokens, and then go into an infinite loop. In this loop, we check for any new retweets from the users that we are monitoring (we could run the retweet task with different users), then update the time everything was last run, and sleep for five minutes.

"We loop through each tweet and get its creation time as a string, which is then converted to a datetime object"

StoreMags.com

# The Code CODE A TWITTER BOT

```
import sys
import time
from datetime import datetime
from twython import Twython

class bot:
    def __init__(self, c_key, c_secret, a_token, a_token_secret):
        # Create a Twython API instance
        self.api = Twython(c_key, c_secret, a_token, a_token_secret)

        # Make sure we are authenticated correctly
        try:
            self.api.verify_credentials()
        except:
            sys.exit("Authentication Failed")
        self.last_ran = datetime.now()

    @staticmethod
    def timestr_to_datetime(timestr):
        # Convert a string like Sat Nov 09 09:29:55 +0000
        # 2013 to a datetime object. Get rid of the timezone
        # and make the year the current one
        timestr = "{0} {0}".format(timestr[:19], datetime.now().year)

        # We now have Sat Nov 09 09:29:55 2013
        return datetime.strptime(timestr, '%a %b %d %H:%M: %S %Y')

    def retweet_task(self, screen_name):
        # Retweets any tweets we've not seen
        # from a user
        print "Checking for new tweets from @{0}".format(screen_name)
```

# The Code CODE A TWITTER BOT

```
        # Get a list of the users latest tweets
        timeline = self.api.get_user_timeline (screen_name = screen_name)
        # Loop through each tweet and check if it was
        # posted since we were last called
        for t in timeline:
            tweet_time = bot.timestr_to_datetime(t['created_at'])
            if tweet_time > self.last_ran:
                print "Retweeting {0}".format(t['id'])
                self.api.retweet(id = t['id'])

if __name__ == "__main__":
    # The consumer keys can be found on your application's
    # Details page located at https://dev.twitter.com/
    # apps(under "OAuth settings")
    c_key=""
    c_secret=""

    # The access tokens can be found on your applications's
    # Details page located at https://dev.twitter.com/apps
    # (located under "Your access token")
    a_token=""
    a_token_secret=""

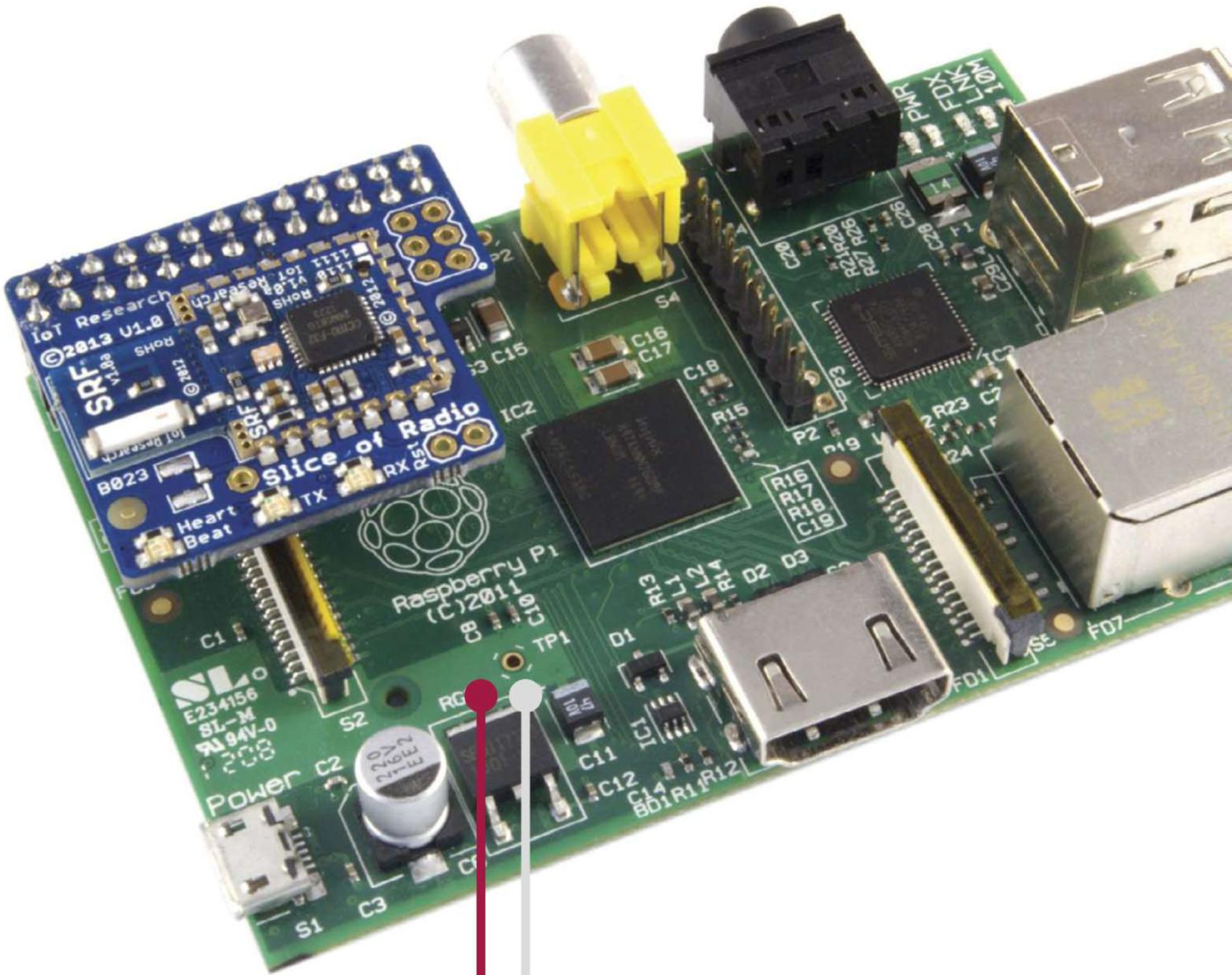
    # Create an instance of the bot class
    twitter = bot(c_key, c_secret, a_token, a_token_secret)

    # Retweet anything new by @LinuxUserMag every 5 minutes
    while True:
        # Update the time after each retweet_task so we're
        # only retweeting new stuff
        twitter.retweet_task("LinuxUserMag")
        twitter.last_ran = datetime.now()
        time.sleep(5 * 60)
```

Storemags.com

# Make a tweeting wireless flood sensor

Flood-proof your basement in just 19 lines of code, or tweak the project to create a custom alarm system...



Flooding saw hundreds of homes right across the world underwater this year, and many would have benefited from having just that little bit extra warning. In order to be better prepared for floods, we're going to show you how you can prototype your own wireless flood sensor in less than ten minutes. Building it might give you just enough warning to dash home from work, move valuable items upstairs and take the lawnmower, caravan and motorbike to higher ground. Handily, it can also be used to detect toilet flushes, water butt levels or any liquid level rise or fall at all – so it's not just something fun to try out, it's practical too!

"Tweepy is an easy-to-use Python library that works great for accessing the Twitter API"

## Sending tweets

Sending a tweet used to be really easy, if a little on the insecure side. These days you need to register an application with your Twitter account – you do have one, don't you? If not, go create one at **www.twitter.com**. At first this project can look a little daunting, however it can be done painlessly in less than five minutes, if you follow these steps closely!

### 01 Link Twitter to mobile

Make sure your Twitter account has a mobile phone number associated with it. In your main Twitter account, click the gears icon at the top-right of the interface and then 'Mobile' in the list. At this stage, just follow the instructions on screen.

### THE PROJECT ESSENTIALS

**Ciseco Raspberry Pi Wireless Inventors Kit**
shop.ciseco.co.uk/raswik

**Float sensor**
shop.ciseco.co.uk/float-switch

**DC power supply between 6v and 12v**

storemags

## 02 Set it all up

With your Twitter username and password, sign in to **https://apps.twitter.com** and click on the button 'Create an application'. In the name field we suggest you use your Twitter account name, add a space and then write 'test'. For the description, just put 'Test poster for Python'. Finally, for the website you can put anything you like. For example, **http://www.mywebsite.com** – but it's important you don't forget the 'http://'.

## 03 Enable reading and writing

Since you want to be able to send tweets, click on the 'Settings' tab, change to 'Read and Write' and then click 'Update'. This might take a few seconds.

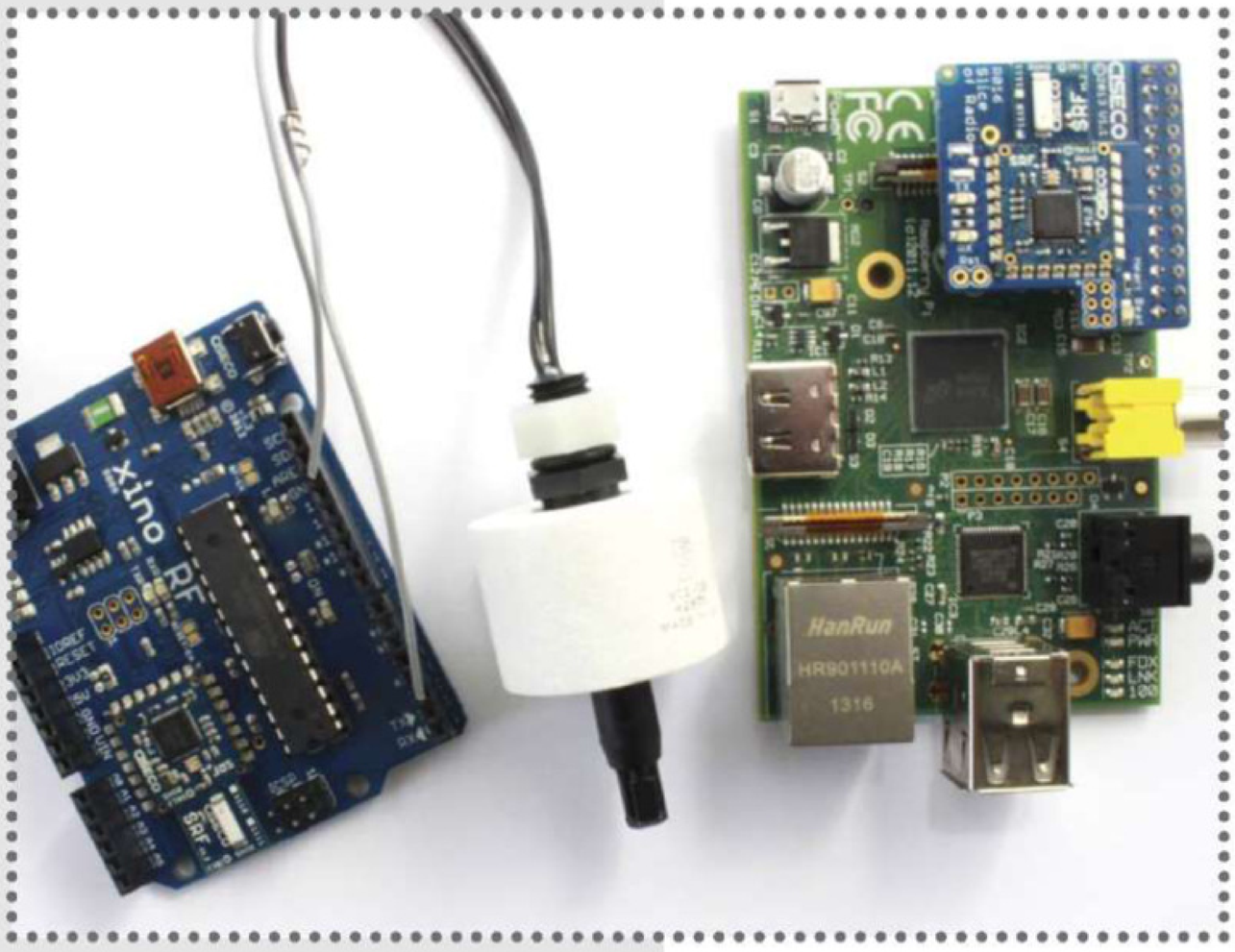
## 04 Generate codes

Now go back to the 'Details' tab. You will see that an 'API key' and 'API secret' are visible, and that there's a 'Create my access token' button. Click that button to obtain all four of the codes you'll need. If you did this before Step 2, or it still says 'Read', all you have to do is click the button to recreate these codes. It really is straightforward.

## 05 Remember the codes

Earlier on 'API' was called 'consumer', and you might have come across this before in examples on the web. We suggest copying the following essentials into Notepad so they don't get lost: API key, API secret, Access token and the Access token secret.

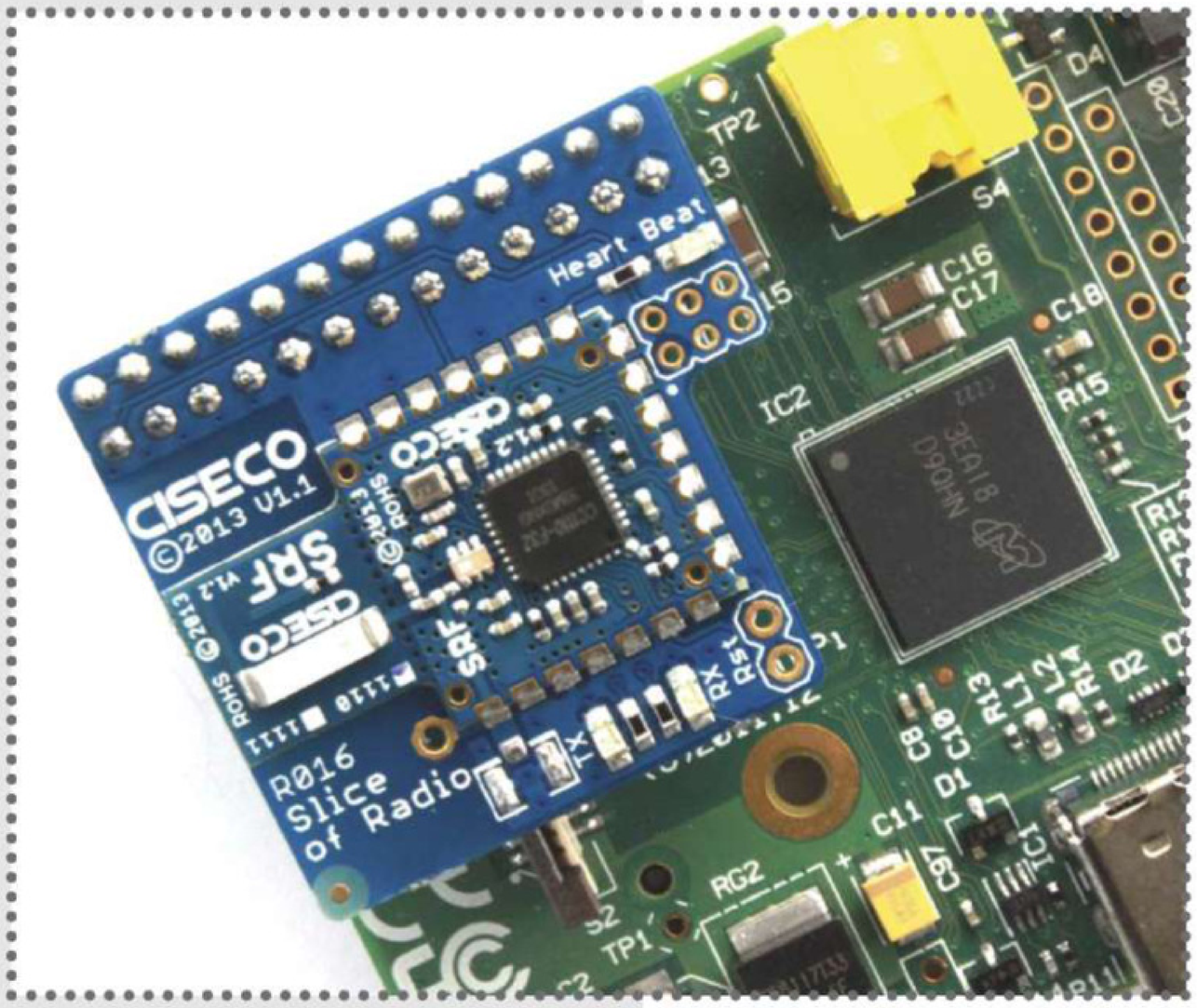
"Copy the following essentials into Notepad so they don't get lost: API key, API secret, Access token and the Access token secret"




**Below** The Wireless Inventors Kit enables you to connect a Pi to an Arduino module from the other side of your house

## No RasWIK?

Not to worry – using different hardware is always a possibility when playing around with the Raspberry Pi. The reason we chose to use the RasWIK is simply because everything except the float switch is in the box and preloaded with software, making it much easier to get up and running quickly. As a bonus addition, this software is also available to download for free.




**Above** The Slice of Radio is a wireless RF transceiver, available from Pimorini: **bit.ly/1A3sI4D**

To build this with a conventional Arduino or clone, you'll need a USB cable and  to leave it 'wired', or use serial-capable radio modules such as the Xbee or APC220. We are, after all, only sending and receiving plain text.

The Arduino sketch used can be downloaded from **http://github.com/CisecoPlc/LLAPSerial**, while the SD image for the OS we used is based on a stock version of Wheezy, which can be downloaded from **bit.ly/SfhLLI**.

### 01 Start simple

To get going with your flood sensor, simply plug in the Slice of Radio to your Pi and insert the preconfigured Raspbian operating system.

### 02 Go to LX terminal

Power up the Raspberry Pi, log in and type STARTX to start the desktop. Double-click the LX Terminal and type the following into the black window:

```
minicom -D /dev/ttyAMA0 -b 9600
```

## 03 Make the connection

Connect the float switch to the XinoRF ground pin (marked GND) and digital I/O pin2. Then, power up the XinoRF (you will see a--STARTED– displayed in minicom)

## 04 Test the sensor sends messages

Wiggle the sensor up and down (you should get a--D02HIGH– when the sensor position is up) and see the RXR LED on the XinoRF flicker with each message sent.

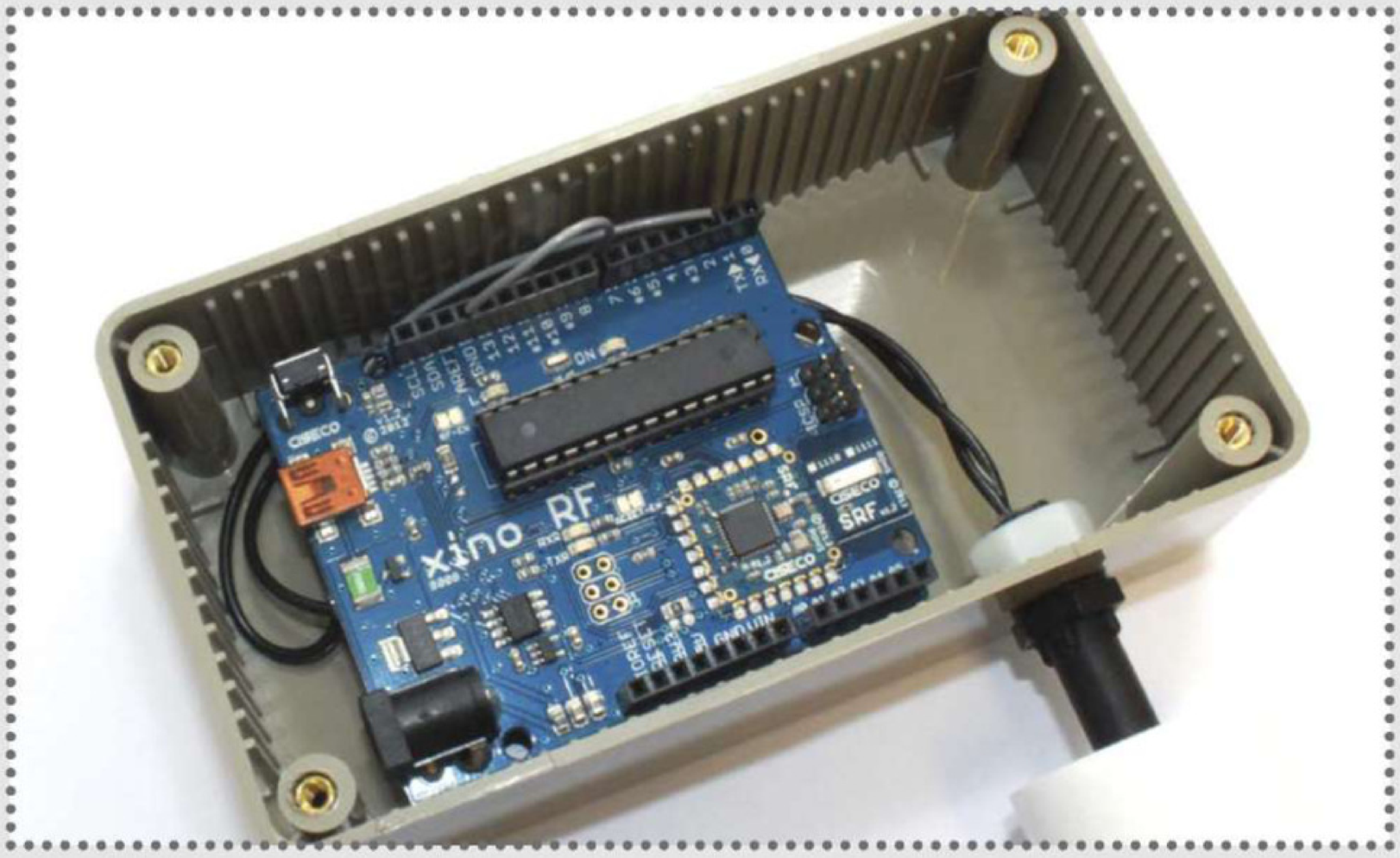
## 05 Install Tweepy

Tweepy is an easy-to-use Python library that works great for accessing the Twitter API. For more information or to check out the documentation, visit **https://pypi.python.org/pypi/tweepy/2.2**. Type in a shell window the following:

```
sudo pip install tweepy
```

## 06 Put the sensor to work

Test your prototype using a regular saucepan of water. If you want to put your flood sensor to real use then place it into a waterproof box and ensure it is mounted securely.




**Left** Also waterproof the hole in the case that your float switch is passing through

"If you want to put your flood sensor to real use then place it into a waterproof box and ensure it is mounted securely"

# The Code TWEETING FLOOD SENSOR

```
#  Let's set up how to talk to Twitter first

import tweepy, serial, time
API_key = "GWip8DF9yf0RYzjIlM6zUg"
API_secret = "Yjipapo56nhkS2SAWtx3M4PAit3HsvWZUYAOghcLn4"
Access_token = "2392807040-19lSoaVOmj8NTvJVteU8x265IPEw2GfY0cS7vuN"
Access_token_secret = "lV4u2ls4oeRZCAxcpaPQNzRDN0lSiUibY0MdCuYKk16Rl"
auth = tweepy.OAuthHandler(API_key, API_secret)
auth.set_access_token(Access_token, Access_token_secret)
api = tweepy.API(auth)

#  Open a com port (yours may differ slightly)

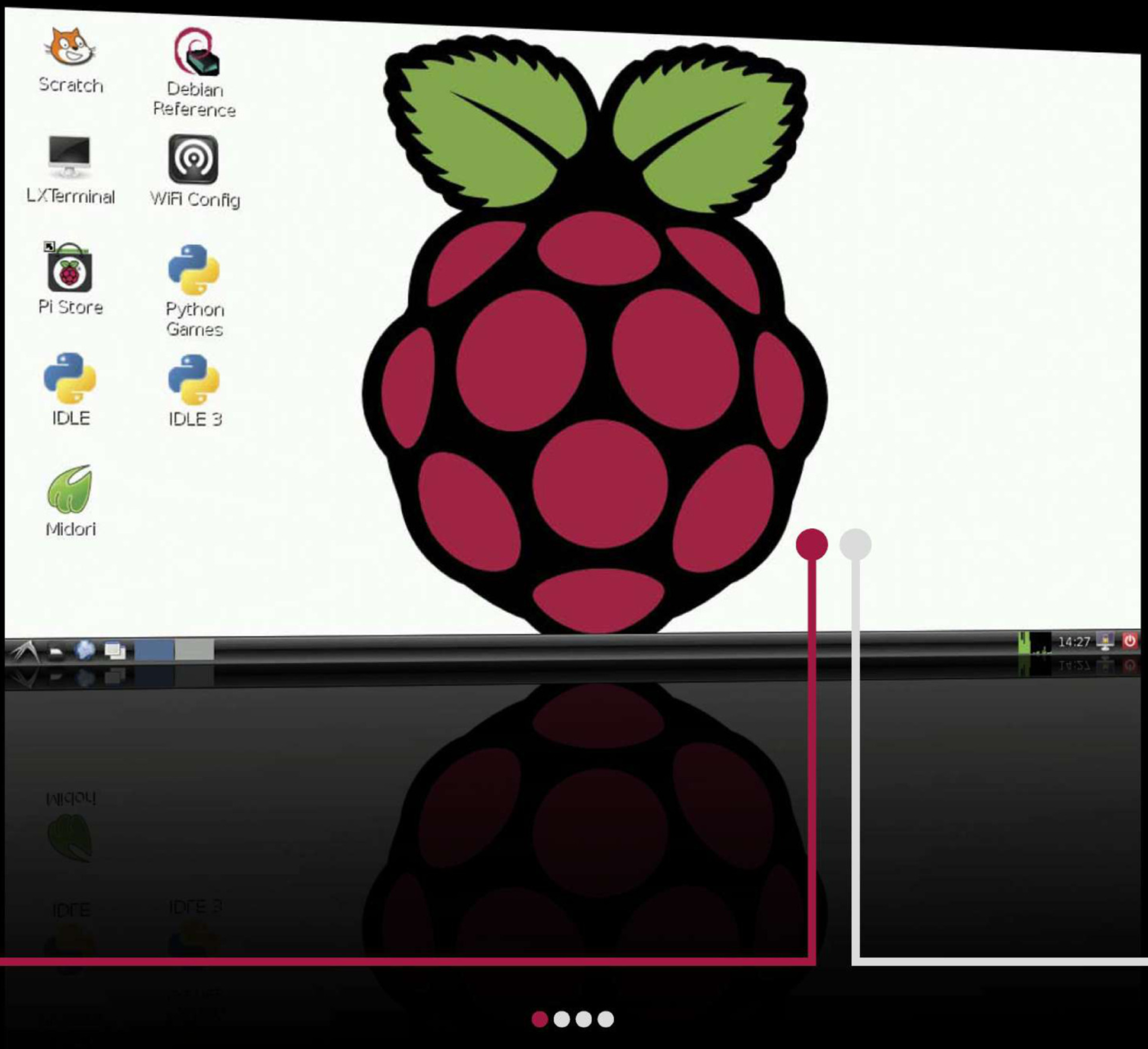
ser = serial.Serial('COM3', 9600)

#  An endless loop checking for sensor alerts

while True:
    SerialInput = ser.read(12)
    if SerialInput == 'a--D02HIGH--':
        TimeNow = datetime.datetime.now()
        DS = TimeNow.strftime('%H:%M:%S %d-%b-%Y')
        AlertText = 'ALERT: LLAP+ device -- flood sensor triggered @ ' + DS
        print (AlertText)
        api.update_status(AlertText)
        time.sleep(10) #stop fast re-triggering
        ser.flushInput()
```

# What is Raspbian?

The operating system that helps to power a Raspberry Pi – but what exactly is Raspbian and what makes it Linux?



StoreMags.com

**Q Raspbian! I use it, you guys use it, I don't think there's any tutorial I've done that doesn't involve using it. What exactly is Raspbian, though? It's Linux, correct?**

**A** Yes! Raspbian is based on Debian, which is a Linux distribution. It's called Raspbian because it's a portmanteau of Raspberry and Debian; much in the same way as RaspBMC or Pidora.

**Q That answers part of my question, but could you elaborate more on the Debian bit in particular?**

**A** Debian is a line of quite popular Linux distributions that is the base of many famous distros; Ubuntu and its offshoots, Tails, SteamOS, wattOS and many more. It's very easy to use and contains a large repository of software, so it's easy to extend and customise it exactly how you want it.

**Q You keep calling it a Linux distribution, which seems oddly specific. What does that mean?**

**A** The different versions of Linux are called distributions or distros. They take the Linux kernel and its other technologies and package it with their own software, software repositories, branding and perhaps even a desktop environment. Debian is an example of one of these distros.

**Q So what sort of software makes it stand out as Debian specifically?**

**A** There's not a huge amount, to be perfectly honest – and neither is that majorly important unless you're into the really low-level security and behind-the-scenes workings. What you will notice is its package management system, Aptitude, which is a major part of Debian and its various flavours and offshoots – if you've used sudo apt-get anywhere in a tutorial, this is it. There's also stuff like the way boot time programs are

## Redesigned Raspbian

Earlier this year a brand new version of Raspbian was released that featured a number of design improvements. These are all thanks to UX designer Simon Long, who implemented changes from the position of the menu bar (now, more logically, at the top of the screen) and highlight colours for menu items when you hover over them, through to a general tidy-up of the menu options and more.




loaded and some necessary tools for compiling software, but you won't often come across it.

**Above** One difference between Raspbian and pure Debian is built-in support for the Pi camera module

### Q So is Linux an operating system?

**A** That's the $64,000 question. The short answer is no but that doesn't really explain why it's not. Linux is a kernel, which, while a core part of any operating system, does not make up the entire operating system. There are various tools that come with the kernel via the GNU project (the reason why it's sometimes called GNU/Linux) which bring it up to a level more akin to an operating system – but it's not quite there yet. You could technically say that these distributions as a product on their own are part of an operating system – eg Raspbian is an OS for the Raspberry Pi – however, in doing so you run into other issues, such as the stuff making up Raspbian being extra components on top of the Linux kernel and GNU tools which wouldn't make up an operating system on its own either. If someone asks you if Raspbian is an operating system, though, just say yes. You'll keep more friends that way.

**Q That's a little bit mind-boggling but I think I understand. So if Raspbian is a version of Debian, is there 'pure' Debian on Raspberry Pi?**

**A** Yes there is. In fact, there was Debian for Raspberry Pi before there was Raspbian. It was the version specifically designed for the CPU that the Raspberry Pi uses, as not every Linux distro has a version that will run on the Raspberry Pi.

**Q Isn't the Raspberry Pi running on ARM? I thought there were other operating systems that ran on ARM?**

**A** Yes, there are ARM versions of Ubuntu and even Windows right now, however they're for different versions of ARM. The Raspberry Pi 1 uses ARM v6 and the Raspberry Pi 2 uses ARMv7. There are also special tools that make better use of the Raspberry Pi hardware and are integrated into the major Raspberry Pi distros.

**Q So why is it that Raspbian has become the default distro for the Raspberry Pi, then?**

**A** It's mostly down to luck and good timing, really. A couple of years ago there was a Fedora spin that was being touted as the official Raspberry Pi distro, however there were some major problems with it. The project that became Raspbian ended up being chosen as a preferred distro around that time and the community really latched onto it. Since then, all the major Raspberry Pi Foundation announcements regarding software have involved Raspbian and pretty much all shared community projects are done on Raspbian, which is why we do a lot of our projects on it as well.

"A couple of years ago there was a Fedora spin that was being touted as the official Raspberry Pi distro, however there were some major problems with it"

storemags.com

StoreMags.com

# Set up a VPN with your Raspberry Pi

The Raspberry Pi is cheap enough to leave on a network you'd like to connect to remotely, so let's learn how to set it up to do just that...

One possible scenario for wanting a cheap server that you can leave somewhere is if you have recently moved away from home and would like to be able to easily access all of the devices on the network at home, in a secure manner. This will enable you to send files directly to computers, diagnose problems and other useful things. You'll also be leaving a powered USB hub connected to the Pi, so that you can tell someone to plug in their flash drive, hard drive etc and you can then put files onto it for them. This way, they can simply come and collect it later whenever the transfer has finished.

We'll be using Arch Linux as the operating system for our VPN server, since it is lightweight and has only the minimum packages required for a working system. This tutorial assumes that you have flashed the latest Arch Linux ARM image to an SD card. If you haven't, the instructions for flashing an image can be found at **linuxuser.co.uk/tutorials/how-to-set-up-raspberry-pi**. You'll only need to go up to the step where you write the image to the SD card, and you'll have to adapt the instructions for using the Arch image rather than Debian.

**THE PROJECT ESSENTIALS**

**A Raspberry Pi**

**Latest Arch Linux image for Pi**
raspberrypi.org/downloads

**A second computer**

## 01 Logging into Arch Linux

Connect the necessary cables to the Pi and wait for the Arch Linux login prompt. The login is 'root', and the password is also 'root'. We'll change the root password from the default later on.

## 02 Run a full system update

Arch Linux runs on a rolling release schedule, meaning that there are no version numbers and software is continually updated. The package manager in Arch

"We'll be using Arch Linux as the operating system for our VPN server, since it is lightweight and has only the minimum packages required"

Storemags.com

Linux is called pacman. Use the command `pacman -Syu` to start a full system update. If for some reason the update fails, try running `pacman -Syu` again. Sadly, the Arch Linux ARM servers tend to be quite busy. There may be a lot of packages to update so it may take a while, especially because the Pi runs from an SD card.

## 03 Install the required packages

Use the command:

```
pacman -S noip netcfg bridge-utils openvpn
```

...to install the required packages mentioned at the start of the article. Answer 'y' to any prompts that you may encounter.

## 04 A word about subnets

One thing to note here is that because we're setting up a client-to-site bridge, we'll be connecting the client to the server's network. This means that the subnet that the server is on needs to be different from the client subnet. For example, the subnet at your advisor's home is 192.168.1.0/24, and the subnet here is 172.17.173.0/24. If the subnet here was 192.168.1.0/24, then there would be a routing conflict because the machine won't know if you're referring to a local address or one on the VPN. It's a good idea to have a non-standard subnet for this reason. In our case, the client subnet is non-standard so it doesn't matter what the server subnet is for now. However, we're still going to change the server subnet at some point because you may end up needing to connect from a network such as public Wi-Fi, which may use a standard subnet. If you need to change your server subnet, we suggest picking a /24 subnet (subnet

> "We'll be connecting the client to the server's network. This means that the subnet that the server is on needs to be different from the client subnet"

storemags

mask 255.255.255.0) in one of the private address ranges:

```
10.0.0.0 to 10.255.255.255
172.16.0.0 to 172.31.255.255
192.168.0.0 to 192.168.255.255
```

You should be able to easily change your network configuration on your wireless router's settings page.

## 05 Investigate your network

We highly recommend assigning a static IP to your server Raspberry Pi rather than being handed one by your router because you'll always know where to find it on the network, which will be useful for accessing it remotely. You'll also need a static IP if you want to access the Raspberry Pi from the Internet. We will first need to find out a couple of things about your current network setup before we start setting a static IP. You can use the commands `ifconfig eth0` and `ip route show` to do this.

## 06 Set up a static IP Address

Now that we have found out things about your network, such as your current IP address, the network mask and so on, we can set up a static IP address. We're going to use the Arch Linux netcfg framework to manage our network connections as we'll need three different connections eventually: Ethernet, which is automatically handled by the bridge adaptor; a VPN tap adaptor; and a bridge adaptor to combine the two. Change directory to the /etc/network.d directory and open a new file called bridge in nano:

```
cd /etc/network.d
nano bridge
```

"We're going to use the Arch Linux netcfg framework to manage our network connections as we'll need three different connections eventually"

Then fill in the bridge configuration to look as follows and save the changes (swapping our network values for your own):

```
INTERFACE=“br0”
CONNECTION=“bridge”
DESCRIPTION=“VPN Bridge connection”
BRIDGE_INTERFACES=“eth0”
IP=‘static’
ADDR=‘192.168.1.215’
NETMASK=‘24’
GATEWAY=‘192.168.1.254’
DNS=(‘192.168.1.254’)
```

Once done, save the file using Ctrl+O followed by Enter, then exit nano using Ctrl+X. We'll add the VPN adaptor to the bridge later on.

We now need to configure what profiles netcfg should load by editing /etc/conf.d/netcfg and configuring the networks array as follows:

```
NETWORKS=(bridge)
```

Save the changes, exit nano and then run the following commands to disable DHCP and enable the Ethernet interface and the bridge with a static IP permanently:

```
systemctl disable dhcpcd@eth0.service
systemctl enable netcfg.service
```

You can now restart your Raspberry Pi for the changes to take effect.

## 07 Log in with SSH

Once the Pi has booted back up, open a terminal on your Linux computer and type ‘ssh root@[IP of your pi]’. Answer yes, to say that you want to connect, and type in the root password, which will still be root. You are now logged in over SSH.

“We need to configure what profiles netcfg should load by editing /etc/conf.d/netcfg and configuring the networks array”

## 08 Change the root password

Since we will probably be exposing an SSH login to the Internet, it is a very good idea to change the password to something much more secure. Type 'passwd', then follow the on-screen instructions to change your password. Your SSH session will stay logged in, but you'll need to use the new password next time you log in. You may also want to change the contents of /etc/hostname to set the hostname to a self-identifying name, such as 'vpnserver' rather than the default 'alarmpi'.

## 09 Set up the public key infrastructure variables

We're going to be using a certificate infrastructure to authenticate OpenVPN. This is where a certificate and private key (which must be kept a secret) is generated for each client, and signed by the certificate authority. Only clients with signed certificates are allowed to connect. The key and certificate are used to encrypt the data sent between the client and the server. This secure approach means that additional username and password authentication on the client is not necessary. There are a bunch of scripts which make setting this up easy. Start by copying the scripts to /etc/openvpn with:

```
cp -r /usr/share/openvpn/easy-rsa/ /etc/openvpn
```

...and then change to that directory.

We're going to be making a template to base our certificates on. Edit the vars file with nano and change the following lines at the bottom of the file from something like:

> "Since we will probably be exposing an SSH login to the Internet, it would be a very good idea to change the password to something much more secure"

```
export KEY_COUNTRY="US"
export KEY_PROVINCE="CA"
export KEY_CITY="SanFrancisco"
export KEY_ORG="Fort-Funston"
export KEY_EMAIL="me@myhost.mydomain"
export KEY_EMAIL=mail@host.domain
export KEY_CN=changeme
export KEY_NAME=changeme
export KEY_OU=changeme
```

...to...

```
export KEY_COUNTRY="UK"
export KEY_PROVINCE=""
export KEY_CITY="Ormskirk"
export KEY_ORG="Home"
export KEY_EMAIL="liam@vpn.home.org"
export KEY_CN="liamvpn-ca"
export KEY_NAME="liamvpn-ca"
export KEY_OU="None"
```

Once you have saved the changes, export the variables with:

```
source ./vars
```

...and then clean any previous configuration with:

```
./clean-all
```

## 10 Create the certificates

Start by generating the certificate authority certificate, with which we will sign everything else (press Enter to leave fields set as they are):

```
./build-ca
```

Now generate a server certificate with:

```
./build-key-server [server hostname]
```

Press Enter when asked for any information, don't fill in a password or company name and accept the

"Only clients with signed certificates are allowed to connect. The key and certificate are used to encrypt the data sent between the client and the server"

request to sign the certificate. Now that's sorted, we need to generate the Diffie–Hellman parameters, needed to allow two users to exchange a secret key over an insecure medium using the Diffie–Hellman key exchange protocol (this may take a minute):

```
./build-dh
```

The final step is to generate a certificate for each client that you would like to connect to the VPN. In this case, our client is a laptop.

```
./build-key liam-laptop
```

Simply do what you did during the build-key-server script and then you'll have all the certificates you need.

## 11 Configure the OpenVPN server

We're going to base our configuration file on the example server configuration file using the command `cp /usr/share/openvpn/examples/server.conf /etc/openvpn/server.conf`. Open /etc/openvpn/server.conf in nano. Start by changing:

```
;dev tap
dev tun
```

…to…

```
dev tap0
;dev tun
```

…because we are using a network tap adaptor, which allows us to bridge the networks rather than create a tunnel. Replace the certificates here with the ones you created:

```
ca ca.crt
cert server.crt
key server.key  # This file should be kept
secret
dh dh1024.pem
```

"Diffie–Hellman parameters are needed to allow two users to exchange a secret key over an insecure medium using the Diffie–Hellman key exchange protocol"

StoreMags.com

In our case, the configuration looked like:

```
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/liamvpn.crt
key /etc/openvpn/easy-rsa/keys/liamvpn.key
dh /etc/openvpn/easy-rsa/keys/dh1024.pem
```

Comment out the line:

```
server 10.8.0.0 255.255.255.0
```

...by placing a semicolon in front of it, because we want an Ethernet bridge rather than a regular server. To enable the Ethernet bridge, uncomment the line:

```
server-bridge 10.8.0.4 255.255.255.0 10.8.0.50 10.8.0.100
```

Then change the values to match your server's network configuration. The first is the server's IP address; the second is the subnet mask. The last two values are the start and end ranges of IP addresses allocatable to connecting clients.

Finally, uncomment the lines:

```
;user nobody
;group nobody
```

to give OpenVPN the least privileges possible and then save the changes to the file.

## 12 Configure the tap interface

Open the file /etc/network.d/tap in nano, add the following lines, and then save the file:

```
INTERFACE='tap0'
CONNECTION='tuntap'
MODE='tap'
USER='nobody'
GROUP='nobody'
```

"The first value is the server's IP address; the second is the subnet mask. The last two values are the start and end ranges of IP addresses allocatable to connecting clients"

We then need to add the tap0 interface to our bridge, so edit /etc/network.d/bridge and change the bridge interfaces line to look like:

```
BRIDGE_INTERFACES="eth0 tap0"
```

Finally, change the networks line in /etc/conf.d/netcfg to:

```
NETWORKS=(tap bridge)
```

Notice that the tap network needs to be started first, so that it can be added to the bridge successfully.

## 13 Enabling OpenVPN

Now that we have configured OpenVPN, we want to enable it permanently. Use the command:

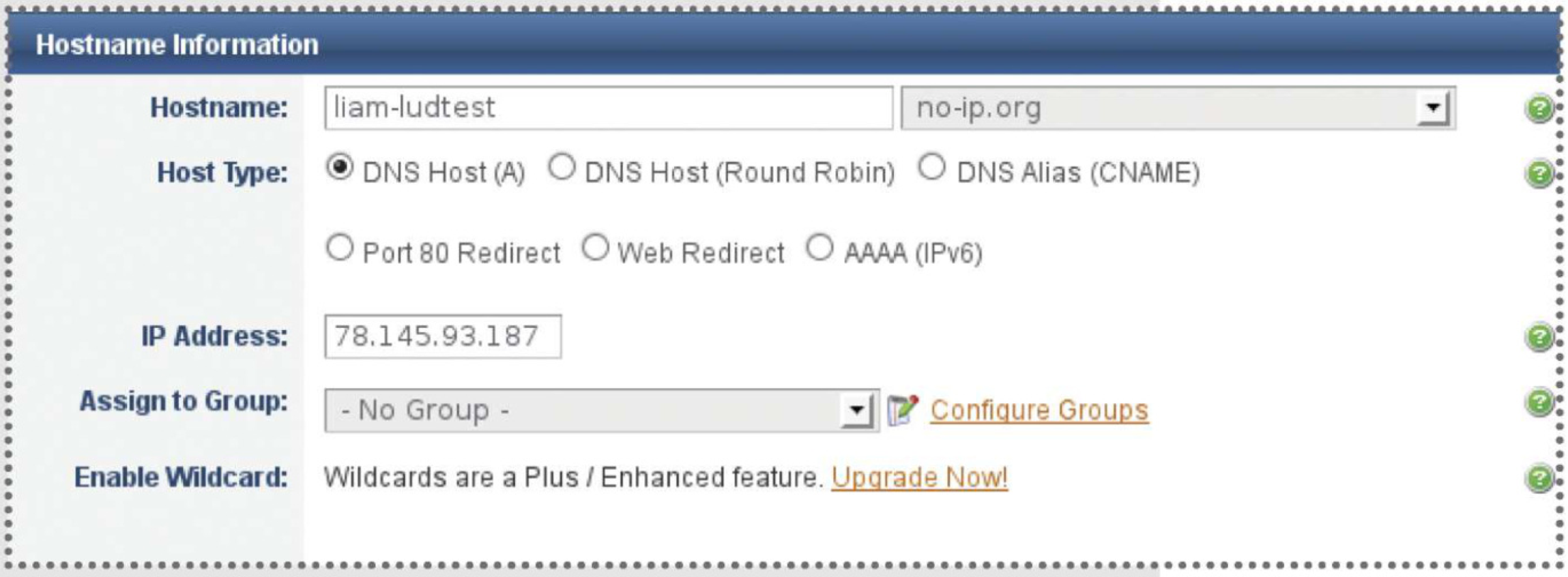
```
systemctl enable openvpn@server
```

...and then reboot the Pi to make sure that everything starts successfully from a clean boot. Our VPN is now configured, so next we're going to set up dynamic DNS and port forwarding so that we can access it from the Internet.

## 14 Set up dynamic DNS

Head over to **no-ip.com/personal** and sign up for the No-IP Free option. Once you have done that, don't bother downloading No-IP's client because we've

**Below** Dynamic DNS provides you with a hostname that's easy to remember and which points to your dynamic IP address



already installed it. Go to your email inbox and follow the activation link that was just sent to you by No-IP. You can now sign into your account. Once you have logged in, select the 'Add a host' option. Choose a hostname and a domain to be part of from the drop-down list. Leave the host type as 'DNS Host' and then click the 'Create Host' button. For example, we used the hostname liam-ludtest with the domain no-ip.org, so we would access that using

```
liam-ludtest.no-ip.org
```

## 15 Configure No-IP

Run the command:

```
noip2 -C -Y
```

…to be taken through interactive configuration of the No-IP client. We left the update interval to the default 30 minutes, meaning the client will check every 30 minutes for an IP address change. Once done, start the daemon with:

```
/etc/rc.d/noip start
```

After a minute or two, your IP address will be accessible via your No-IP hostname. However, it's likely that trying it from inside your house will simply take you to your router's homepage.

## 16 NAT port forwarding

It is likely that there are multiple devices behind your router that all use the same external IP address. This is because of the shortage of IPv4 addresses, and also because it is more secure to segregate the Internet from your internal home network. NAT (network address translation) forwards a port from

> "Multiple devices behind your router use the same external IP address because of the shortage of IPv4 addresses, and because it is more secure"

the router's external IP address to a computer on the LAN (local area network). In this case, we'll want to forward any traffic for TCP port 22 that comes to your router's external IP address to the IP address of your Raspberry Pi. TCP port 22 is the port used for SSH. SSH will provide remote access to your Raspberry Pi, and also access to any files on it via SCP (Secure Copy Protocol). You'll also want to forward UDP port 1194, as that's what OpenVPN uses.

The configuration of port forwarding really depends on the router that you are using, so you may have to look it up. The chances are that it will be hidden away in the 'Advanced' section of your wireless router. You should be able to access your router by typing your No-IP hostname into your web browser. If not, it should be at the address of your default gateway that we used earlier on.

On our router, we had to go to Advanced>NAT> Port Mapping and then add a mapping, as shown below. We then had to add a second mapping for OpenVPN, using port 1194 specifying UDP rather than TCP as the protocol.

**Below** The exact interface will be different depending on your router, but will look something like this.

Port Mapping

New | Remove | Help

| Interface | Protocol | Remote Host | External Port | Internal Port | Internal Host | Mapping Name | Enable | Remove |
|---|---|---|---|---|---|---|---|---|
| nas_0_38 | TCP | | 22 | 22 | 192.168.1.191 | SecureShellServer(SSH) | Enable | ☐ |

Settings

Type: ◉ Customization ○ Application Please Choose ...

Interface: nas_0_38 Protocol: TCP

External port: 22 Internal port: 22

Internal host: 192.168.1.191 Remote host:

Mapping name: SecureShellServer(SSH)

## 17 Install an OpenVPN client

We'll use a virtual machine running Ubuntu 12.04 as our example VPN client. There are simply too many possible combinations to show them all. There are a couple of options that must be used on every client, however:

- Use a TAP device
- Use LZO data compression
- Do not use the default gateway on the remote network (on Ubuntu, this is called 'Use this connection only for resources on its network'). This basically means 'don't tunnel my Internet through this VPN'. If this option is disabled, then the client's Internet connection wouldn't work because we haven't configured our VPN to deal with Internet.

Ubuntu uses Network Manager to configure its networks, so the instructions we give here should be almost identical to any other distro that uses the same thing. Ubuntu doesn't come with the OpenVPN plug-in for Network Manager by default, so we'll need to start by installing it. From a terminal, run:

```
sudo apt-get update
sudo apt-get install network-manager-openvpn-gnome
```

## 18 Copy the required certificates to the client

We need three files from the Raspberry Pi to be able to connect successfully:

- The certificate authority certificate
- The client certificate
- The client key

"Ubuntu uses Network Manager to configure its networks, so the instructions we give here should be almost identical to any other distro that uses the same thing"

We'll be using SCP to copy the files into the /etc/openvpn/keys directory:

```
cd /etc/openvpn
sudo mkdir keys
cd keys
sudo scp root@[Pi's IP address]:/etc/openvpn/
easy-rsa/keys/ca.crt .
sudo scp root@[Pi's IP address]:/etc/openvpn/
easy-rsa/keys/[client].crt .
sudo scp root@[Pi's IP address]:/etc/openvpn/
easy-rsa/keys/[client].key .
sudo chmod +r *
```

Note that we use chmod to add read permissions because the files need to be readable by all users. We need to do this because the Network Manager GUI doesn't run as root.

## 19 Create the VPN connection

Note that you'll probably want to be on a different subnet to your server, otherwise it's likely you'll run into connectivity issues on the client because of the aforementioned routing problem. We managed to work around this problem while at home by using a virtual machine that's connected with NAT. As far as the virtual machine is concerned, it's on the 10.0.2.0/24 subnet.

Click on the Network icon in the top menu bar and click on the 'Edit connections' option. You will then be shown a window that has multiple tabs at the top. Go to the VPN tab and click 'add'. Select OpenVPN as the connection type and then click on 'Create'. Now fill in the appropriate information.

"You'll probably want to be on a different subnet to your server, otherwise it's likely you'll run into connectivity issues on the client"

## 20 Advanced settings

We need to set the advanced settings that we mentioned before:

- Use a TAP device
- Use LZO data compression

## 21 Route settings

The final thing we need to set is the option to 'Use this connection only for resources on its network'. To do this, go to the IPv4 Settings tab and click the Routes button. Tick the box for the aforementioned option and then click Okay. Once you have done this, you can Save your connection and close the Network Connections window.

## 22 Test your connection

Click on the Network icon in the menu, hover over the VPN Connections option and then click on the VPN that you just created. You should see a success message and a padlock as part of the Network icon. Open up a terminal and run ifconfig to check that the tap device has been corrected with an appropriate IP address, and also that you can ping a device behind the VPN.

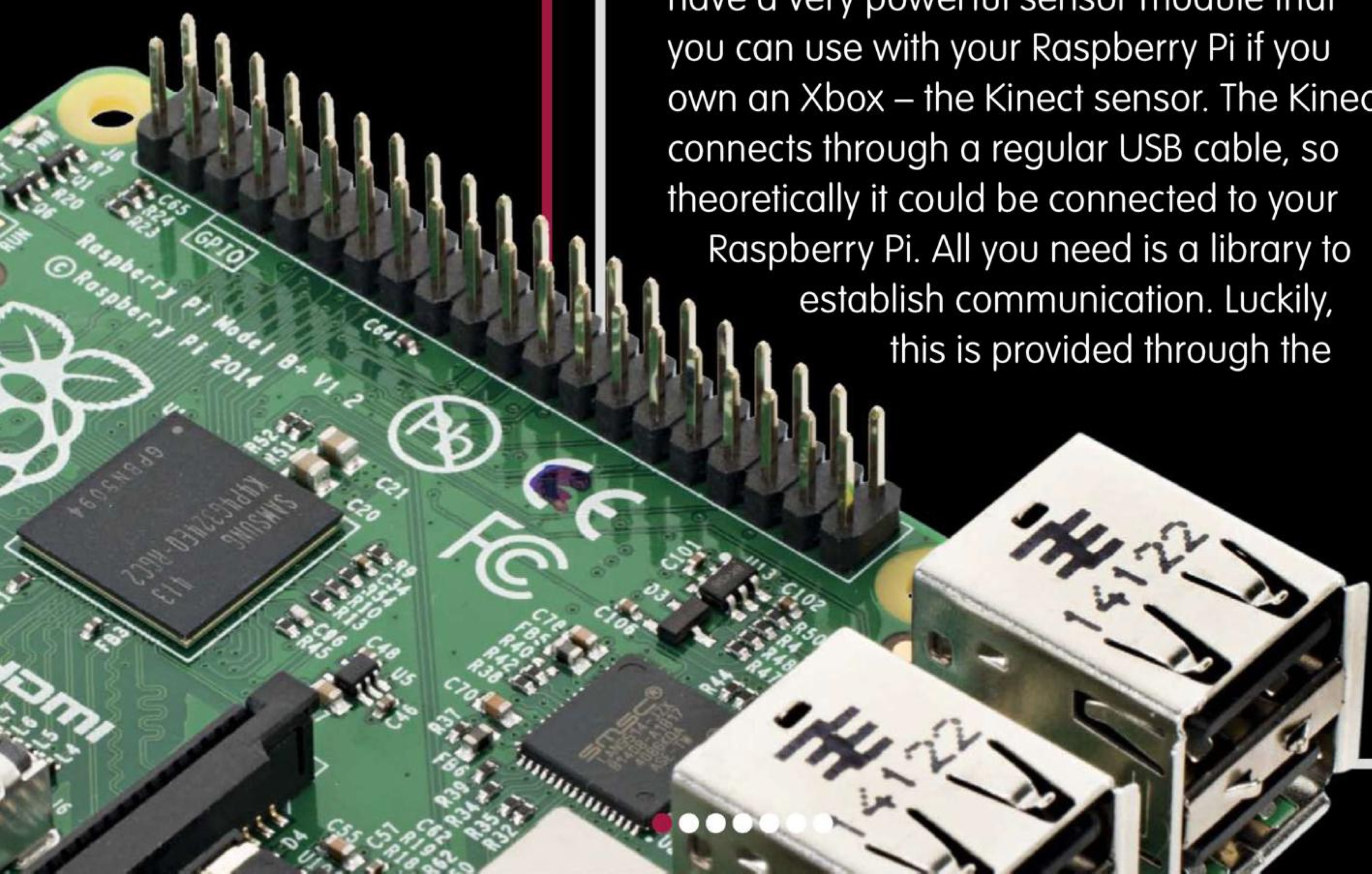
"Run ifconfig to check that the tap device has been corrected with an appropriate IP address"

StoreMags.com

# Add vision to your Raspberry Pi

The Kinect can provide a very robust vision system to your Raspberry Pi – and all through Python!

"The Kinect connects through a regular USB cable, so theoretically it could be connected to your Raspberry Pi"



If you do even a cursory search online, you will be able to find any number of modules that will add functionality to your Raspberry Pi. Many of these are sensor modules that let you observe the world around you. What you may not know is that you probably already have a very powerful sensor module that you can use with your Raspberry Pi if you own an Xbox – the Kinect sensor. The Kinect connects through a regular USB cable, so theoretically it could be connected to your Raspberry Pi. All you need is a library to establish communication. Luckily, this is provided through the

OpenKinect project (**www.openkinect.org**). If you use Raspbian, then you should have everything you need available in the package repository. The base libraries are available with the command:

```
sudo apt-get install freenect
```

If you were to write C code, you could stop here. Since we are focusing on Python, you will also need to install the Python wrappers. You do this with the command:

```
sudo apt-get install python-freenect
```

The last step to getting set up is to connect the Kinect to your Raspberry Pi. The USB ports on the Raspberry Pi probably can't provide enough power for the Kinect, so you will need a powered USB hub placed between the two devices.

Now that everything is connected together, we will look at some Python code to start communicating with the sensor. We will be borrowing heavily from the demos included in the Python wrapper source code. The first step is to import the freenect module, with:

```
import freenect
```

Be sure that everything is connected and powered up before doing this. You should now have access to your Kinect. The first thing you can do, just to verify that you do indeed have control of your Kinect, is to set the LED value. You can set it to green, red, yellow, blinking or off, with the function…

```
freenect.set_led(dev, led)
```

"The USB ports on the Raspberry Pi probably can't provide enough power for the Kinect, so you will need a powered USB hub"




**Above** Many people may not realise that they can harness the power of the Xbox's sensor with Python

…where led is an integer from 0 to 6 representing the various options. The parameter dev contains a pointer to the Kinect device. You can also tilt the Kinect by using the function set_tilt_degs(dev, tilt_degrees). Just be sure to keep it between 0 and 30 degrees. In order to get the device parameter, you need to get the freenect module to tell you what it is.

One way to do this is to use the runloop wrapper function. This function takes a function you wrote as the body, initialises the Kinect and runs the body in an infinite loop. But what if you want to break out of the loop? You can use a global variable to trigger what to do. But how can you set it? Here we'll need to introduce signals.

Signals are messages that are sent to your program by the kernel. For example, if you enter Ctrl+C, this sends the signal SIGINT, or the interrupt signal, to your program. Other examples are SIGKILL (the kill signal), or SIGTERM (the terminate signal). In Python, you can work with signals by importing the signal module. You can then set up a handler to listen for particular signals and do something when you get one. In the example code, we have a global variable named keep_running that tells freenect whether to continue in the runloop function or not. You can create a handler function that looks like the following:

```
def handler(signum, frame):
  global keep_running
  keep_running = False
```

Once you have this function, you can register it to be fired when a particular signal is received. So, if you wanted to fire this when the user types Ctrl+C, you would use the following command:

```
signal.signal(signal.SIGINT, handler)
```

Now we should start trying to get some data from the Kinect. There are both synchronous and asynchronous versions of the get functions that can achieve this for us. The asynchronous versions require callback functions, so we will skip these versions for now. The synchronous versions block on each call until the data is returned. On the return, you will get a list of objects and you will be interested in the first element. This looks something like:

```
freenect.sync_get_video()[0]
```

The simplest thing to do is to display this video data. Since we are in Python, we can use matplotlib and get an animated display. A simplified loop looks like...

```
image_rgb = matplotlib.imshow(get_video(),
interpolation='nearest', animated=True)
while keep_running:
  image_rgb.set_data(get_video())
  matplotlib.draw()
```

...where the function get_video() wraps the call to freenect's sync_get_video. Again, we can use the signal

"Synchronous versions of the get function block on each call until the data is returned. On the return, you will get a list of objects"

handler we defined above to be able to let the user break out of this infinite loop. Things get more interesting when you start to analyse all of this video data – and the OpenCV project provides tools to do some pretty serious data analysis. In order to do this, you will need to do some data conversion first. The main difference is that the data from the Kinect is in RGB order, while OpenCV is expecting it to be in BGR order. Once you make this conversion, you can reproduce the display from above with…

```
import cv
cv.NamedWindow('Video')
while 1:
  cv.ShowImage('Video', get_video())
  if cv.WaitKey(10) == 27:
    break
```

…where get_video() is a wrapper that takes care of the conversion for OpenCV. Along with the video information, there are also equivalent functions to get depth information, and the last source of information available from the Kinect is accelerometer data. With this data, you can figure out the orientation of the Kinect to help you map out where the video data is located relative to the Kinect sensor. The get_accel() function returns a set of x, y and z values for the x, y and z parts of the local acceleration. With all of this data, you can start some pretty interesting projects.

There is not a lot of documentation available on how to use the Python wrapper for freenect, so you will need to poke around a bit to discover exactly what you can do. When you learn something new, be generous and share it with others so that everyone can improve – there are a lot of exciting things to discover when you combine a Kinect with your Raspberry Pi!

"You can figure out the orientation of the Kinect to help you map out where the video data is located relative to the Kinect sensor"

# The Code FULL CODE LISTING

```
# Need to import the required modules
import freenect
import time
import random
import signal

# Some global variables
keep_running = True
last_time = 0

# This is the function that will be
# run as the body of a run loop in freenect
# This example will randomly change the
# LED and the tilt angle of the kinect
def body(dev, ctx):
    global last_time
    if not keep_running:
        raise freenect.Kill
    if time.time() - last_time < 3:
        return
    last_time = time.time()
    led = random.randint(0, 6)
    tilt = random.randint(0, 30)
    freenect.set_led(dev, led)
    freenect.set_tilt_degs(dev, tilt)

# We will need a signal handler to let
# the user interrupt the run loop
def handler(signum, frame):
    """Sets up the kill handler, catches SIGINT"""
    global keep_running
    keep_running = False
```

# The Code FULL CODE LISTING

```
# Now we set the signal handler and start the
# run loop - it will stop when you press CTRL-C
signal.signal(signal.SIGINT, handler)
freenect.runloop(body=body)

# If we want to get video data and display
# it, we need to import matplotlib
import matplotlib.pyplot as mp
keep_running = True

# Need to set up matplotlib
mp.ion()
mp.gray()

# Get an initial frame and set up the signal handler
image_rgb = mp.imshow(freenect.sync_get_video()[0], interpoloation='nearest',
animated=True)
signal.signal(signal.SIGINT, handler)

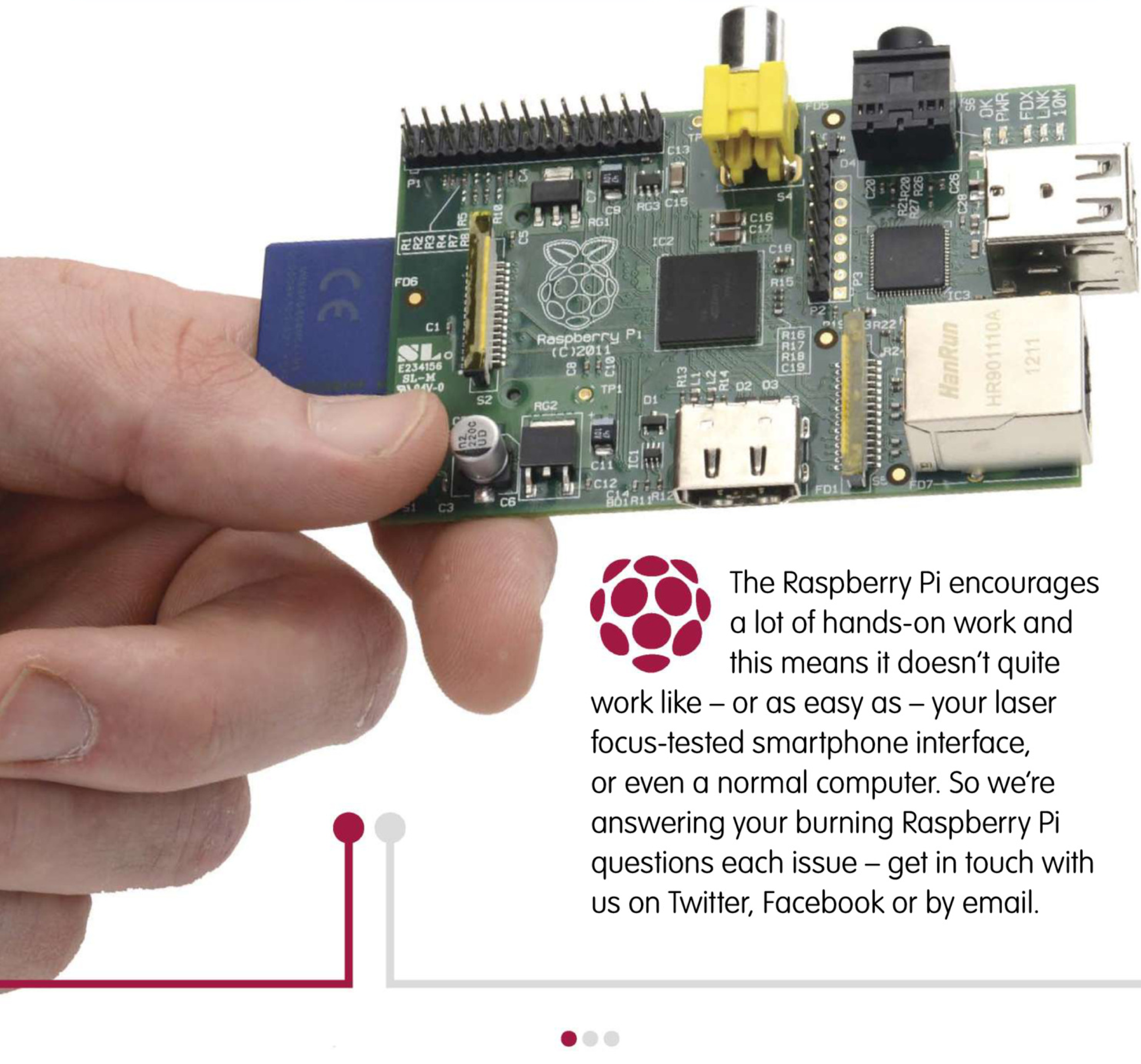
# And now we loop
while keep_running:
    image_rgb.set_data(freenect.sync_get_video()[0])
    mp.draw()
    mp.waitforbuttonpress(0.01)
```

"If you enter Ctrl+C, this sends the signal SIGINT, or the interrupt signal, to your program"

# Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@imagine-publishing.co.uk




The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easy as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

**Should I upgrade to the Raspberry Pi 2?**
**Tyler via email**

The Raspberry Pi 2 is indeed a massive upgrade – at least in terms of power – to the original Raspberry Pi models. Does that make it worth the replacement for an original Raspberry Pi? Well, the old models will still be fully supported for the time being so it's not a necessary upgrade, especially if your setup is doing just fine for you. On the other hand, if you could use a little extra power and a bit of future-proofing, it is only £25. We think it's definitely a worthwhile upgrade.



**Will my code work on the Raspberry Pi 2?**
**Shelby via Facebook**

As the Raspberry Pi Foundation has been keen to reiterate, all projects, code, programs and such will work on the Raspberry Pi 2. They want to make the switchover as painless as possible, and all you really need to do is copy the files from your old SD card onto your new one. It's as simple as that.

**Follow @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag**

## Win a new Raspberry Pi 2

Last month, the Raspberry Pi Foundation announced the brand new Raspberry Pi 2. Available to buy now and shipping with an updated version of Raspbian, the Pi 2 is the next evolution of the standard Model B.

The key difference is the brand new BCM2836 processor, which uses an ARMv7 core and is now six times more powerful than the old Pi. >>

There's a capacitor that fell off my Pi! Can I fix it?
**Riley via Twitter**

Yep, this is actually a very fixable problem and not at all uncommon! With a sharp eye, a steady hand and a bit of solder you can easily reattach it to the board. If you're not so confident in your soldering skills, then you don't even need to worry about it – the Raspberry Pi should work just fine without it.

Are there any keyboard/mouse combos I can use to save USB space?
**Chase via Facebook**

There are plenty of great, portable and wireless keyboards that also include a touchpad with them. The Kano's magic wand is a great example of this, with the keyboard on one side and the mousepad on the other, however there are plenty of alternatives you can find browsing online. If you're not in the market for a Kano (**kano.me**), we recommend checking out the wireless, hand-held eSynic KB05: **bit.ly/17h8F8v**. Most of these types of device use a bluetooth dongle as well, so you can use one without being limited by the cable – useful for Pi's kept in tricky places.




## **Win** a new Raspberry Pi 2

The Pi 2 will work with all your existing projects and is fully backwards compatible on the software side, so it's the perfect upgrade

We have five of these brilliant boards to give away, and for a chance to win one of them all you need to do is pay a visit to the webpage below and then answer one simple question.

Enter the Raspberry Pi 2 competition:
**bit.ly/1CGPjUE**

# Next issue

**Get inspired** **Expert advice** **Easy-to-follow guides**

## Build a smart ALARM CLOCK



**Plus** Science on a Raspberry Pi

Get this issue's source code at:
www.linuxuser.co.uk/raspicode